PHP 5

LE GUIDE COMPLET

3ème Édition

Maîtrisez PHP 5 de A à Z !

Micro Application

François-Xavier Bois

1ère Édition - Mai 2008

Auteurs François-Xavier BOIS

ISBN : 978-2-300-014147

MICRO APPLICATION
20-22, rue des Petits-Hôtels
75010 PARIS
Tél. : 01 53 34 20 20
Fax : 01 53 34 20 00
http://www.microapp.com

Support technique :
Également disponible sur
www.microapp.com

Retrouvez des informations sur cet ouvrage !

Rendez-vous sur le site Internet de Micro Application **www.microapp.com**. Dans le module de recherche, sur la page d'accueil du site, entrez la référence à 4 chiffres indiquée sur le présent livre.
Vous accédez directement à sa fiche produit.

→ RECHERCHE
• PAR MOTS CLÉS
1414
OK

Avant-propos

Destinée aussi bien aux débutants qu'aux utilisateurs initiés, la collection *Guide Complet* repose sur une méthode essentiellement pratique. Les explications, données dans un langage clair et précis, s'appuient sur de courts exemples. En fin de chaque chapitre, découvrez, en fonction du sujet, des exercices, une check-list ou une série de FAQ pour répondre à vos questions.

Vous trouverez dans cette collection les principaux thèmes de l'univers informatique : matériel, bureautique, programmation, nouvelles technologies...

Conventions typographiques

Afin de faciliter la compréhension des techniques décrites, nous avons adopté les conventions typographiques suivantes :

- **gras** : menu, commande, boîte de dialogue, bouton, onglet.
- *italique* : zone de texte, liste déroulante, case à cocher, bouton radio.
- `Police bâton` : Instruction, listing, adresse internet, texte à saisir.
- ✂ : indique un retour à la ligne volontaire dû aux contraintes de la mise en page.



Il s'agit d'informations supplémentaires relatives au sujet traité.



Met l'accent sur un point important, souvent d'ordre technique qu'il ne faut négliger à aucun prix.



Propose conseils et trucs pratiques.



Donne en quelques lignes la définition d'un terme technique ou d'une abréviation.

Chapitre 1 Introduction 13

1.1. **Les langages de programmation** ... 14
1.2. **Le PHP** ... 20
1.3. **Internet, comment ça marche ?** ... 31
1.4. **Check-list** ... 46

Chapitre 2 L'environnement de travail 47

2.1. **WampServer** ... 48
Installation ... 48
Premiers pas ... 53
Le menu de Wamp ... 56
L'éditeur Notepad++ ... 59
2.2. **Paramétrage de PHP** ... 60
2.3. **Check-list** ... 64

Chapitre 3 Les fondamentaux 65

3.1. **Structure d'un programme** ... 67
3.2. **Les commentaires** ... 72
3.3. **Les variables** ... 74
3.4. **Les constantes** ... 78
3.5. **Les types de données** ... 80
Les données numériques ... 80
Les chaînes de caractères ... 82
Le type NULL ... 85
Changement de type ... 85
3.6. **Les structures de contrôle** ... 86
Les conditions ... 87
Les boucles ... 93
3.7. **Organisation du code** ... 99
Les fonctions ... 99
Inclusion de fichier ... 109
3.8. **Check-list** ... 113

Chapitre 4 Les tableaux 115

4.1. **Présentation** ... 116
Les tableaux scalaires ... 117
Les tableaux associatifs ... 118
Les tableaux multidimensionnels ... 119
4.2. **Parcours d'un tableau** ... 121
Boucle foreach ... 121
Utilisation du pointeur interne ... 123
Utilisation des références ... 124
4.3. **Les fonctions** ... 125
Suppression d'une cellule ... 125

Affichage d’un tableau 126
Taille d’un tableau 127
Conversion chaînes / tableaux 128
Adjonction, soustraction d’éléments 130
Tri 131
Présence d’une valeur dans un tableau 134
Sérialisation 134
4.4. Les opérateurs sur les tableaux 136
4.5. Check-list 137

Chapitre 5 Dates et heures 139

5.1. La notion de timestamp 140
Création d’un timestamp 141
Conversion 142
Comparaison de dates 144
5.2. Formatage d’une date 146
Echappement de caractères 149
Constantes 150
5.3. Contrôle de validité d’une date 152
5.4. Check-list 153

Chapitre 6 Les formulaires et transmissions de données 155

6.1. Qu’est-ce qu’un formulaire ? 156
6.2. Les différents widgets 158
INPUT TEXT 159
TEXTAREA 161
SELECT 162
INPUT CHECKBOX 164
INPUT RADIO 164
INPUT BUTTON 165
INPUT HIDDEN 166
6.3. Passer des paramètres à un script PHP 166
La variable $_GET 167
Query String 174
La méthode POST 177
Le mode register_globals on 180
6.4. Check-list 181

Chapitre 7 En tête HTTP et authentification 183

7.1. Requêtes et réponses 184
Extension LiveHTTPHeaders 184
La requête 186
La réponse 187
7.2. Fonction header() 188
7.3. Page d’erreur 190

7.4. Authentification 192
7.5. En bref 196

Chapitre 8 JavaScript, contrôle de formulaires et AJAX 197

8.1. Présentation de JavaScript 198
Les fonctions 199
L'interaction avec les widgets 205
La bibliothèque Prototype 214
8.2. Des vérifications simples en PHP 216
8.3. Les expressions régulières 222
8.4. Ajax 226
AJAX et Prototype 226
Échange de données au format JSON 229
8.5. Check-list 234

Chapitre 9 L'envoi d'un formulaire par courriel 235

9.1. Configuration requise 236
9.2. Mail Texte 237
9.3. Mail HTML 242
9.4. Check-list 248

Chapitre 10 L'enregistrement dans une base de données 249

10.1. Les bases de données 250
Qu'est ce qu'un SGBD ? 250
Organisation d'un SGBD 253
Les requêtes 254
10.2. PHP et MySQL 259
Premières requêtes 259
Enregistrement d'une fiche 272
10.3. Envoi de fichier 277
Modification de la structure d'une table 277
Envoi de fichier 278
10.4. Le couteau suisse du développeur web : phpMyAdmin 283
10.5. Check-list 289

Chapitre 11 La gestion d'une base de données 291

11.1. L'authentification 292
11.2. La mise à jour d'une table 296
L'instruction input hidden 300
La commande UPDATE 300
11.3. La suppression : DELETE 308
11.4. La factorisation du code 314
La fonction include 315

L'amélioration visuelle : les CSS 325
11.5. Recherche et tri au sein d'une base 331
Définir la fonction de recherche 331
Définir la fonction de tri 334
11.6. Check-list 337

Chapitre 12 La gestion des fichiers 339

12.1. Manipuler des fichiers 340
Les fichiers de cache 340
L'écriture 342
La lecture 344
Les fichiers modèles : templates 347
12.2. Créer des fichiers spéciaux 351
Les fichiers compressés 351
Les fichiers Excel 356
Les fichiers Flash 358
Les fichiers PDF 363
Les fichiers image 365
12.3. Check-list 382

Chapitre 13 La programmation objet 383

13.1. Classes et objets 385
Classes 385
Objets 387
Conversion 390
Constructeur et destructeur 391
13.2. Les méthodes magiques 393
__sleep() et __wakeup() 393
__toString() 394
Surcharge des accesseurs 394
13.3. Polymorphisme 398
Principe général 398
Visibilité 399
13.4. Les interfaces 401
13.5. Itérateurs 403
13.6. Exceptions 405
Principe général 405
La classe Exception 407
13.7. Réflexion 409
13.8. Version objet de la génération de graphique 410
13.9. Check-list 416

Chapitre 14 XML 417

14.1. Le format 418

14.2. **SimpleXML** ... **421**
Création ... 422
Lecture ... 425
14.3. **Formats spéciaux** ... **426**
RSS ... 427
XHTML ... 434
SVG ... 435
14.4. **Check-list** ... **437**

Chapitre 15 Les cookies et les sessions 439

15.1. **Les cookies** ... **440**
Aspects techniques ... 441
Application : la mini-boutique FoxShop ... 444
15.2. **Les sessions** ... **472**
15.3. **Check-list** ... **482**

Chapitre 16 La gestion de la sécurité 483

16.1. **La sécurité avec PHP** ... **485**
Le b-a ba ... 485
Mise à jour de PHP ... 486
Initialiser toutes les variables ... 486
Utiliser les constantes ... 487
Se méfier de la puissance de certaines fonctions ... 488
Dangers de la fonction mail ... 489
Les cookies et les sessions ... 490
Les transferts de fichiers ... 491
Inclusion de fichier ... 492
16.2. **Sécuriser les bases de données** ... **493**
Les injections SQL ... 493
Les Cross Site Scripting ... 494
16.3. **Sécuriser le serveur web** ... **496**
Les directives PHP ... 497
Les directives Apache ... 500
La sécurité HTTPS ... 503
16.4. **Les outils d’analyse** ... **503**
16.5. **Check-list** ... **504**

Chapitre 17 Les trucs et astuces 505

17.1. **PHP** ... **506**
Définir autrement une chaîne de caractères ... 506
Raccourcir un if... else... ... 507
L’autre syntaxe des structures de contrôle ... 508
Raccourcir un simple bloc echo ... 509
Donner une valeur par défaut à un paramètre d’une fonction .. 510
Transmettre un nombre variable de paramètres à une fonction . 511

Utiliser un opérateur de comparaison de type 512
Les attributs __FILE__ et __LINE__ 513
Les variables variables 514
Les opérateurs sur les tableaux 514
Les techniques d'optimisation en PHP 515
Les fonctions include() et require() 516
L'affichage tampon : output buffering 518
Fin de bloc PHP 519
Le paramètre caché de break et continue 519
Chaîne de caractères sous forme de tableau de caractères 520
Rendre disponible un site wamp sur internet 521
17.2. MySQL 523
Récupérer un enregistrement de manière aléatoire 523
Optimiser ses tables 524
Autres optimisations 525
17.3. HTML et Javascript 525
Empêcher l'autocomplétion 525
Définir le rafraîchissement automatique d'une page 526

Chapitre 18 Les fonctions PHP 527

18.1. Les fonctions mathématiques 529
18.2. Les chaînes de caractères 538
18.3. Les expressions régulières 559
18.4. Les tableaux 561
18.5. Les fonctions de dates et d'heures 583
18.6. Les fichiers et les répertoires 588
18.7. L'interface avec MySQL 611
18.8. Les images 622
18.9. Les variables 638
18.10. La configuration PHP 642
18.11. Fonctions diverses 645

Chapitre 19 Annexes 649

19.1. Webographie 650
PHP 650
MySQL 652
Apache 652
Internet et le Web 653
Et les blogs 654
Divers 654
19.2. PHP 654
Les opérateurs 654
Les variables prédéfinies 657
Les mots réservés 668
Les différences entre PHP 3 et PHP 4 669
Les différences entre PHP 4 et PHP 5 672

19.3. **MySQL** **673**
Les types 673
Les fonctions 674
19.4. **Les caractères HTML spéciaux** **679**
19.5. **Les feuilles de styles : CSS** **683**

Chapitre 20 Index 693

Dédicace

À Laurence et Marie-Castille.

Chapitre 1

Introduction

Les langages de programmation 14
Le PHP 20
Internet, comment ça marche ? 31
Check-list 46

Tout en étant consacré à un langage de programmation aussi pointu qu'avancé, cet ouvrage reste destiné à un large public. Quelques connaissances élémentaires dans le domaine du Web (HTML) mises à part, aucune compétence informatique particulière n'est indispensable à la compréhension des différents sujets abordés au sein du présent ouvrage. Il est de ce fait particulièrement destiné aux web designers et aux webmestres, aux étudiants et, plus généralement, à toute personne aspirant à aller plus loin dans la création de sites et d'applications web.

Tout au long des chapitres, nous nous attacherons à illustrer les différents concepts étudiés à l'aide d'exemples que nous enrichirons au fur et à mesure des chapitres. Nous étudierons aussi bien les bases du langage (syntaxe, variables, fonctions) que certains aspects plus avancés (envoi de courriels, manipulation de fichiers, interaction avec les bases de données, gestion des cookies et sessions, programmation objet).

Nous profiterons également de certains chapitres pour découvrir certaines technologies adjacentes à PHP : le HTML (les formulaires), le SQL (le langage des bases de données), le XML (un format universel d'échanges de données) et le langage Javascript (qui a vu sa réhabilitation récente avec l'émergence du concept de Web 2.0).

Dans le cadre de ce premier chapitre, nous nous intéresserons tout d'abord aux différents langages de programmation pour nous concentrer ensuite sur le langage PHP, son histoire, son mode de fonctionnement, ses avantages et ses défauts.

Nous profiterons aussi de ce chapitre introductif pour réaliser un rapide tour d'horizon du Web et d'Internet en général.

1.1. Les langages de programmation

PHP est un langage de programmation. Il permet d'écrire des programmes, tout comme les mathématiques permettent de résoudre des problèmes. Très en vogue actuellement, il est cependant loin d'être le seul dans sa catégorie. Plusieurs centaines de langages ont ainsi vu le jour depuis la naissance de l'informatique dans les années 1950. Parmi les plus connus peuvent être cités les langages C, C++, C#, Java, Perl, Python, Basic, ActionScript, etc.

Un programme informatique est composé de lignes d'instructions ; l'ensemble de ces lignes forme le code source (ou listing) du

programme. Dans la vie courante, les instructions suivantes pourraient être apparentées à un programme :

Listing 1-1 : un programme dans la vie courante

```
1- insérer la carte
2- composer le code secret
3- renouveler l'étape 2 en cas d'échec
4- composer le montant
5- appuyer sur le bouton validez
6- retirer les billets
7- récupérer la carte
```

Pour un même objectif, le code source d'un programme est différent selon le langage utilisé.

Listing 1-2 : Programme écrit en PHP

```
for ($i = 1; $i <= 10; $i++)
{
  echo "i = $i\n";
}
```

Listing 1-3 : Programme écrit en C

```
for (i = 1; i <= 10; i++)
{
  printf("i = %d\n",i);
}
```

Listing 1-4 : Programme écrit en Java

```
for (i = 1; i <= 10; i++)
{
  System.out.println("i = " + i);
}
```

Listing 1-5 : Programme écrit en Python

```
for i in range(1,11):
     print "i = ", i
```

Listing 1-6 : Programme écrit en Perl

```
for ($i = 1; $i <= 10; $i++)
{
  print "i = $i\n";
}
```

Ces exemples prouvent qu'en dehors de quelques différences d'ordre syntaxique tous ces langages de haut niveau sont extrêmement similaires. Depuis les origines de la programmation, les concepteurs de langage de haut niveau se sont systématiquement « empruntés » les bonnes idées, tout en mettant de côté les faiblesses et les limitations.

Certains langages récents trouvent ainsi leurs origines dans les années 1960. En analysant minutieusement sa syntaxe, nous pouvons ainsi nous rendre compte que le langage de Microsoft C# (C Sharp) est directement issu d'un langage aujourd'hui complètement oublié : le BCPL.

Vous devriez donc être en mesure, à l'issue de cet ouvrage, de lire sans difficulté majeure un listing de code écrit dans la plupart des langages modernes de haut niveau.

Langages interprétés et langages compilés

Il serait bien évidemment inutile et fastidieux de connaître plusieurs langages si ces derniers proposaient tous les mêmes fonctionnalités. Comme vous pouvez vous en douter, ce n'est pas du tout le cas : chaque langage dispose de ses particularités, de ses avantages et de ses défauts.

La première grande différence à observer entre les différents langages cités précédemment se situe dans leur mode de fonctionnement : certains sont à classer parmi les langages interprétés et d'autres parmi les langages compilés.

Tableau 1.1 : *Langages interprétés et langages compilés*

Langages interprétés	Langages compilés
PHP	C
Perl	C++
Python	Java
JavaScript	Pascal



Détails

Rien n'étant jamais simple en informatique, vous pourrez effectivement lire qu'il est possible de compiler du Perl ou du PHP, que les exécutables Java et C# sont en fait « interprétés » par des machines virtuelles, que des processeurs Crusoe de Transmeta deviennent des interpréteurs de binaires. Ce ne sont toutefois que des détails, et il est évident qu'aujourd'hui toutes ces notions ont tendance à se mêler les unes aux autres.

La différence entre ces deux types de langages se situe au niveau de leur mode d'exécution.

Pour exécuter un programme écrit dans un langage non interprété, il est nécessaire de compiler le code source pour en faire un binaire. Le compilateur est le programme qui se charge de cette opération. Chaque langage dispose ainsi d'un compilateur qui lui est propre : celui du C s'appelle `gcc`, celui du Java se nomme `javac`. À l'issue de cette phase dite de compilation le binaire pourra être exécuté par la machine. Le compilateur s'est en réalité chargé de convertir toutes les lignes de code dans un langage de bas niveau (l'assembleur dans le cas du C), illisible par l'homme mais « compréhensible » par un processeur.

Listing 1-7 : exemple de code écrit en assembleur

```
.data

msg:
  .ascii  "Hello, world!\n"
  len = . - msg

.text


    .global _start


_start:

  movl  $len,%edx
  movl  $msg,%ecx
  movl  $1,%ebx
  movl  $4,%eax
  int  $0x80

  movl  $0,%ebx
  movl  $1,%eax
  int  $0x80
```

Pour exécuter un script écrit avec un langage interprété, il faut, comme son nom l'indique, passer par un interpréteur. Cet interpréteur lit le code pas à pas et le convertit au fur et à mesure en instructions pouvant être traitées par le processeur. Quand on parle de PHP, on parle donc à la fois du langage et de l'interpréteur.



Script

Le terme « script » est souvent utilisé lorsque l'on souhaite faire référence à un programme écrit dans un langage interprété. On parle ainsi de script PHP ou Perl.

Voyons rapidement les avantages et les inconvénients de chacun de ces deux modes.

Avantage du langage compilé

La conversion en binaire est réalisée une fois pour toutes lors de la phase de compilation. La suite Office de Microsoft, par exemple, a été compilée une fois, et ce sont des versions binaires que l'on trouve sur les étalages des grandes surfaces. Un programme compilé est donc plus rapide à s'exécuter qu'un programme interprété qui, lui, devra être converti à chaque exécution.

Inconvénient du langage compilé

Le binaire issu de la compilation n'est pas exécutable « universellement ». Ainsi, un binaire exécutable sur un PC fonctionnant sous Windows ne le sera pas sur un PC sous Linux ou OS2 : il s'agit ici d'une incompatibilité de système d'exploitation. De la même manière, un binaire compilé sur PC ne pourra pas fonctionner sur Mac ou Sun : il s'agit alors d'une incompatibilité d'architecture machine. Il est donc facile d'imaginer le casse-tête pour des sociétés souhaitant faire fonctionner et vendre leur logiciel sur le plus grand nombre de plateformes possible.

Avantage du langage interprété

Un programme PHP n'étant ni plus ni moins qu'un simple fichier texte contenant des lignes de code, il est interprétable sur tout type de machine ou de système d'exploitation sans que cela nécessite la modification de la moindre virgule : on appelle cela la « portabilité ». Le marché potentiel d'un programme écrit en PHP est par là même bien plus vaste que celui d'un programme compilé qui, généralement, n'est développé que pour un système d'exploitation et une architecture donnés.

Inconvénient du langage interprété

En plus de la relative lenteur par rapport au langage compilé, il convient de noter un inconvénient de taille pour les personnes souhaitant vendre leur programme : la fourniture du code source. Alors qu'il est impossible de deviner comment un programme compilé a été conçu, il est tout à fait possible pour une société cliente ayant acheté un

programme écrit en PHP de voir comment celui-ci a été codé et ainsi de voler les idées et le savoir-faire du concepteur.

Autre problème de taille : la nécessité de disposer de l'interpréteur pour pouvoir exécuter un script. Alors que vous pouvez transmettre un binaire par courriel et être sûr qu'il pourra être exécuté chez votre ami, il conviendra pour un script PHP de vérifier que cet ami dispose préalablement sur son ordinateur de l'interpréteur PHP. Or, il peut être assez gênant d'imposer l'installation d'un tel environnement pour la simple exécution d'un programme.



Compilation de PHP

La société Zend, dont nous allons parler plus loin dans ce chapitre, a développé un outil qui permet de convertir un programme PHP en un fichier contenant un code intermédiaire illisible par l'humain, mais lisible par un interpréteur PHP (et cela quels que soient l'architecture et le système d'exploitation !). Il ne s'agit ni plus ni moins que d'un compilateur PHP déguisé.

Les domaines d'application

En plus de cette différence de fonctionnement, les langages ont souvent été conçus pour des domaines d'application précis.

- ASP, PHP, CFM : le Web.
- C : applications système.
- Java : applications pour systèmes embarqués (téléphones portables, cartes à puce).
- Perl : administration système.
- C++ : applications avec interfaces graphiques.

Bien que tout programme puisse être écrit avec tout langage, certains vous permettront de le développer en 10 lignes, alors que d'autres en nécessiteront 300.



Choix du langage

Bien que PHP soit un langage aussi polyvalent qu'attractif, il ne faut surtout pas tomber dans l'excès qui consisterait à vouloir tout réaliser en PHP. D'autres langages, pour des problématiques bien précises, peuvent se révéler supérieurs à PHP. Il est donc toujours intéressant de se tenir



informé et de surveiller les autres technologies et nouveautés (cela étant d'autant plus vrai en informatique où les choses évoluent beaucoup plus vite qu'ailleurs).

1.2. Le PHP

Le PHP est un langage interprété qui a été conçu dès son origine pour le Web. Il est aujourd'hui devenu le leader incontesté dans ce domaine. Plus de 9 millions de sites l'ont aujourd'hui choisi comme plateforme de développement web.

Les raisons du succès

Elles sont à la fois nombreuses et variées.

Rapidité, stabilité, scalabilité, sécurité

PHP est le langage de *scripting* le plus rapide du marché. C'est réellement important quand vous devez réaliser un site devant recevoir plusieurs centaines de milliers de visiteurs par jour. Plus le script met de temps à être interprété, plus l'attente est importante pour l'internaute. Or, n'oubliez jamais que rien n'est pire sur le Web que de faire attendre un internaute !

Cette rapidité est d'autant plus impressionnante que PHP dispose d'autres propriétés toutes aussi essentielles.

- Stabilité : PHP n'est pas « buggé » et ne « plante » pas.
- Scalabilité : qu'il y ait cent ou un million d'internautes qui viennent sur votre site, PHP continuera à exécuter vos scripts (certes plus lentement dans le cas d'un million de requêtes).
- Sécurité : PHP est un système très sûr dont les rares failles ont toujours été corrigées dans la journée.



Sécurité et PHP

Il faut bien faire la différence entre la sécurité de PHP en tant que système et la sécurité d'un logiciel écrit en PHP. Le fait que PHP soit sécurisé n'implique pas pour autant qu'une application écrite en PHP soit elle-même sécurisée. Un programmeur peut ainsi tout à fait laisser dans son code une faille de sécurité qui pourra être exploitée par un



pirate. Il est donc très important de prendre de bonnes habitudes en vérifiant toujours que son code ne contient pas de faiblesse. Le chapitre consacré à la sécurité devrait vous y aider.

Open Source

Le projet PHP est un projet open source. L'open source est un mouvement planétaire qui regroupe les meilleurs développeurs mondiaux et qui a pour principe fondateur la mise à disposition des sources des logiciels (c'est-à-dire les listings de code qui ont permis de réaliser le logiciel). Ainsi, alors que l'ASP n'est développé que par Microsoft, PHP est un projet sur lequel travaillent des centaines d'étudiants, de chercheurs et ingénieurs à travers le monde. En disposant des sources, tout un chacun peut étudier la manière avec laquelle le langage est conçu et peut aussi corriger les éventuels dysfonctionnements (bugs). Cela explique directement que PHP soit un langage extrêmement stable ne souffrant que de très rares bugs ou failles.

Appartenant à tout le monde et à personne en même temps, les logiciels Open Source ont un énorme avantage par rapport à leurs cousins propriétaires : ils ne peuvent pas disparaître. Si une société éditrice d'un langage dépose le bilan, le langage disparaît avec la société. Pour PHP, cela ne peut arriver. N'importe quel étudiant dispose des sources et peut reprendre le flambeau. En ces temps de troubles et de difficultés pour les éditeurs de logiciels, cette notion ne doit pas être mise de côté.

Un autre avantage à travailler avec des logiciels open source tel que PHP est d'avoir à sa disposition une énorme bibliothèque de scripts dont les sources peuvent être récupérées gratuitement sur le Web. Il devient aujourd'hui assez rare de ne pas trouver sur le Web un morceau de code qui ne répondrait pas exactement à vos besoins.



Le chapitre « Webographie » vous indique une série de sites proposant le téléchargement de scripts PHP.

Fonctionnalités

Grâce à l'open source, chacun peut ajouter sa pierre à l'édifice, en améliorant ou en développant certaines parties. Le cycle de

développement de PHP est par conséquent très rapide, et chaque nouvelle version est accompagnée de son lot de nouvelles fonctionnalités. PHP contient donc un très grand nombre d'extensions qui permettent par exemple :

- de générer des images, des fichiers PDF, Flash ;
- de se connecter à des serveurs FTP, LDAP, de mail ;
- de travailler avec des bases de données (MySQL, MS SQL, Oracle, Informix, PostgreSQL) ;
- de manipuler des fichiers XML ;
- d'interagir avec des Web Services ;
- de s'interfacer avec des systèmes de paiement sécurisé.

PHP est un langage d'une très grande flexibilité. Quelle que soit la complexité du logiciel à concevoir, il est très peu probable de se retrouver limité par PHP. Comme le C, PHP vous permet de tout faire, le plus souvent très rapidement. Cette ressemblance avec le C ne s'arrête d'ailleurs pas là. PHP dispose en effet d'une syntaxe très proche de celle du C. Quand on sait que le C est un des langages les plus répandus, cela se révèle un choix tactique : beaucoup de programmeurs ont pu de la sorte passer du C à PHP en quelques heures et venir enrichir la communauté de développeurs. Plus un langage dispose de développeurs, plus vous avez de chances d'obtenir des réponses dans les forums, de trouver des documents et des exemples sur le Web.

Gratuité

PHP fait partie de cette famille de logiciels que l'on qualifie de *free software*, *free* dans le sens de « libre » (open source), mais également dans le sens de « gratuit ». Bien que PHP soit de loin ce qui se fait de mieux dans le domaine, il est, à la différence de ses principaux concurrents (ASP, ColdFusion…), entièrement gratuit. PHP n'est pas le seul logiciel gratuit et open source dont nous allons parler dans ce livre : Linux (système d'exploitation), Apache (serveur web), MySQL (moteur de base de données) sont d'autres logiciels incontournables du monde du Web et sont tout aussi libres et gratuits.

Universel

L'interpréteur PHP est aujourd'hui disponible sur un très grand nombre d'architectures (PC, Mac), de systèmes d'exploitation (Windows, Mac OS X, Linux, Unix, etc.) et de serveurs web (Apache, IIS, AOLserver,

Roxen, etc.). Ainsi, si vous changez un jour d'hébergeur, il y a de fortes chances que votre application continue de fonctionner.

Apache/Linux

Bien que fonctionnant sur la grande majorité des serveurs web et sur la plupart des systèmes d'exploitation, PHP est avant tout lié au serveur Apache et au système Linux. Ce sont là les véritables applications phares du monde de l'open source et du Web. Apache est de loin le serveur web le plus utilisé au monde. C'est lui qui vous sert les pages des plus gros sites mondiaux (Yahoo!, Google). Linux est quant à lui le deuxième système d'exploitation derrière Windows dans le domaine des serveurs web.

Les concurrents

PHP est loin d'être le seul langage de scripting pour le Web. On trouve parmi ses concurrents…

Ruby

Ses avantages :

- Il s'agit d'un véritable langage objet où tout élément du langage est lui-même objet. Il ravira les développeurs exigeants au niveau modélisation ainsi que ceux, plus débutants, souhaitant mettre en place des interfaces graphiques le plus rapidement et simplement possible.
- La disponibilité de la plateforme RAILS pour réaliser des applicatifs web en AJAX est un véritable atout du fait de la très grande popularité de cet environnement.
- Gratuit, open source et disponible sur une grande variété de plateformes.

Ses inconvénients :

- Langage assez récent, il est encore peu répandu chez les hébergeurs et risque de mettre encore quelques années avant d'être accepté au sein des grands groupes.

Python

Ses avantages :

- Ce langage est extrêmement bien pensé et permet une qualité de développement objet largement supérieure à celle du PHP.
- Gratuit, libre et largement portable.
- Le langage Python tend de plus en plus à remplacer Perl dans le cœur des administrateurs système et voit sa base d'utilisateurs s'étendre de jour en jour.

Ses inconvénients :

- La syntaxe du langage fondée sur l'indentation peut paraître douteuse à certains.
- Le Web est loin d'être la priorité des concepteurs.

ASP (Microsoft)

Ses avantages :

- ASP est ce qu'il y a de mieux lorsque l'on souhaite ne travailler qu'avec des outils Microsoft et être assuré de la compatibilité avec IIS, Front Page, Visual Studio, SQL Server, Access.
- Les outils clients sont généralement très bien réalisés, que ce soit pour gérer les pages, la base de données ou le serveur web. Cela permet à un non-ingénieur système d'administrer une solution web complète.

Ses inconvénients :

- ASP souffre de faibles performances et ne peut être exécuté qu'avec IIS sous Windows.
- Le serveur web IIS, pierre angulaire d'une solution Microsoft, n'est pas à citer en exemple en termes de sécurité. Les attaques gravissimes sur des machines disposant de ce logiciel sont communes (voir Red Code, Nimda) et ont parfois paralysé des sociétés entières.
- Il s'agit d'un choix onéreux dans la mesure où le logiciel est payant tout comme les technologies adjacentes (Visual Studio, Front Page, SQL Server) qui sont vivement recommandées afin de rester dans un environnement Microsoft et d'éviter les incompatibilités. Il s'agit ici de la partie émergée de l'iceberg car il convient d'ajouter à ces licences un prix d'hébergement et d'administration souvent beaucoup plus élevé. Et ne croyez pas faire des économies en hébergeant en interne car il s'agira dans ce

cas d'investir dans une machine disposant d'énormes ressources autant au niveau du processeur que de la mémoire.

ASPX, C# (Microsoft)

Ses avantages :

- À la différence d'ASP, les scripts ASPX peuvent maintenant être exécutés sur des serveurs Apache disposant du module libre et gratuit Mono (www.mono-project.com/Main_Page).
- Les ASPX peuvent être écrits en C# qui est sans nul doute un magnifique langage de programmation.

Ses inconvénients sont les mêmes que l'ASP.

CFM (Macromedia-Allaire) : ColdFusion

Ses avantages :

- L'environnement de développement de ColdFusion est ce qui peut se faire de mieux dans le genre. Vous disposez en achetant ce logiciel d'un outil central disposant d'une interface graphique complète vous permettant de développer votre code, vos pages web, d'envoyer vos documents sur un serveur FTP.
- Le langage a été développé avec la simplicité en ligne de mire. Par conséquent, il s'agit peut-être de la meilleure solution pour un public débutant ne souhaitant pas aller très loin dans le développement web.

Ses inconvénients :

- Le langage est plutôt lourd, mal conçu et n'évolue que très lentement. Dès que l'on souhaite aller assez loin dans le développement, les défauts et les limitations apparaissent très vite (on pense notamment à la gestion assez primaire des « expressions régulières »).
- Les hébergeurs proposant le ColdFusion ne sont pas nombreux et font souvent payer ce service assez cher. Le serveur web est aussi propriétaire et, même si ses performances sont honorables, elles sont loin d'être aussi bonnes que celles d'Apache.
- Comme l'ASP, cet environnement de développement est payant.

Perl

Ses avantages :

- Très vieux langage, Perl dispose d'une bibliothèque d'extensions extrêmement riche (par exemple, création de fichiers Excel à la volée, connexion à une multitude de serveurs, etc.).
- Gratuit, open source et disponible sur une grande quantité de plateformes.

Ses inconvénients :

- Ce langage n'a pas été développé dans une optique web et peut donc souffrir d'une certaine lourdeur.
- L'installation sur une machine cliente est souvent bien plus compliquée que les systèmes vus précédemment.

Des logiciels en ligne

De plus en plus d'applicatifs sont développés avec ces langages orientés web. Comme il est nécessaire d'être connecté, on les qualifie fréquemment de logiciels « en ligne » (*online softwares* ou *web applications*).

Les avantages de tels applicatifs sont assez nombreux :

- L'applicatif étant centralisé, sa mise à jour devient extrêmement simple (nul besoin de changer quoi que ce soit sur les postes des utilisateurs).
- Le marché est immense. Toutes les personnes disposant d'un navigateur web et d'une connexion au Net peuvent y accéder (quel que soit le système d'exploitation ou l'architecture).
- Les données de la société sont centralisées, tous les employés peuvent y accéder de manière collaborative.
- Seul le serveur web doit être sécurisé, ce qui simplifie grandement la tâche des responsables informatiques. De la même manière, les sauvegardes deviennent très simples à gérer.

Tout naturellement, ces logiciels se sont développés principalement pour le monde de l'entreprise : gestion commerciale et financière, intranet et extranet, commerce électronique, gestionnaire de planning, de messagerie, d'agenda, etc.

Microsoft a bien compris que le marché du logiciel allait dans ce sens, et c'est dans cette optique qu'il tend à pousser sa plateforme de développement .NET. L'avenir ira sans nul doute vers une décentralisation et une location des logiciels.



Interactions entre langages

Parmi les extensions peu connues de PHP, nous pouvons mentionner le fait que PHP est désormais capable de « récupérer » du code d'autres langages et de l'exécuter. Plus exactement, PHP est en mesure de charger des objets écrits en Java ou en C# et de faire appel aux différentes fonctions et méthodes contenues dans ces mêmes objets. Cette fonctionnalité est surtout utile dans le monde de l'entreprise où d'énormes librairies extrêmement complexes ont déjà été écrites (notamment en Java) pour des environnements peu ou pas documentés. En permettant ces interactions, PHP trouve sa place au cœur des grands chantiers informatiques et commence à être envisagé par les SSII souvent soucieuses d'aller au plus vite, au plus simple et au plus sûr.

L'histoire

Le langage PHP, comme la plupart des grands projets open source, est né d'une volonté individuelle et isolée. Conçu au départ pour récupérer des informations sur les internautes qui visitaient sa page personnelle, PHP est resté pendant plus d'une année le jouet de son unique concepteur : Rasmus Lerdorf. À cette époque, PHP signifiait *Personal Home Page* (on en parlait fréquemment sous le nom PHP/FI).



Figure 1.1 :
Le logo de PHP

Ce n'est donc qu'en 1995, à la suite d'une annonce dans les newsgroups, que ce projet est devenu accessible au reste du monde. PHP devenait alors *PHP: Hypertext Preprocessor*. Très simple au début, le langage a pu, avec l'aide de la communauté open source, s'enrichir de nouvelles fonctionnalités, notamment l'accessibilité aux bases de données.

Dès cette époque, le projet s'internationalisa, et des développeurs, originaires de Norvège, d'Israël, d'Allemagne, des États-Unis, prirent l'habitude de participer régulièrement au développement de PHP.

De 15 000 en 1996, le nombre de sites utilisant PHP a atteint 50 000 en 1997. C'est à cette période que le projet est passé sous la direction de Zeev Suraski et Andi Gutmans. Ces deux étudiants israéliens décidèrent de réécrire le langage de A à Z. Cette nouvelle version devint PHP 3.

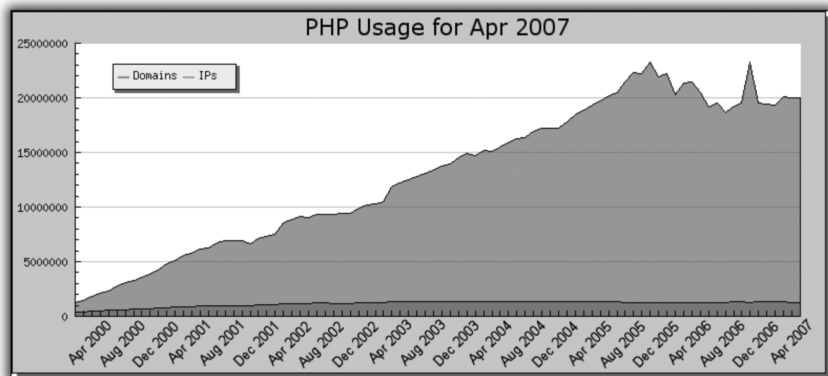


Figure 1.2 : *Évolution du nombre de sites qui utilisent PHP*

Après cet énorme bond en avant que fut PHP 3, les développeurs se remirent au travail avec la ferme intention d'optimiser le cœur de PHP pour obtenir des performances encore meilleures (personne ne se plaignait pourtant des performances de PHP à cette époque !). Plutôt que d'exécuter les instructions une à une, il fut décidé de choisir une nouvelle approche : « compiler » le code, puis l'exécuter. Ce travail a eu lieu dans le cadre du Zend Engine, qui est aujourd'hui le véritable cœur de PHP 4. Cette nouvelle mouture est sortie en 2000. L'intégration du Zend Engine dans le cadre de PHP 4, outre le gain en performances, apporte une plus grande modularité et une meilleure extensibilité. PHP est aussi devenu indépendant de la couche « serveur web », ce qui en fait aujourd'hui un des systèmes les plus compatibles du marché.

PHP 5, sorti en juin 2004, est basé sur un Zend Engine de nouvelle génération (version 2). Cette nouvelle version vise principalement à combler des lacunes du langage dans les domaines de la programmation objet, des bases de données et des services web. En s'attaquant à ces domaines, PHP se donne une légitimité accrue dans le monde de l'entreprise et peut d'autant plus facilement être envisagé pour des

projets critiques. PHP 4 reste aujourd'hui la version de PHP la plus populaire du fait de sa stabilité extrême et de sa très large diffusion parmi les hébergeurs.

Alors que nous écrivons ces lignes, PHP 6 est en phase de développement. L'objectif principal de cette version sera de rendre PHP parfaitement compatible avec les rouages complexes de l'Internationalisation (I18N).

Aujourd'hui

Zeev Suraski a fondé la société Zend Technologies (www.zend.com). Cette société fournit des logiciels payants autour du langage PHP :

- Zend Encoder rend vos scripts PHP illisibles (ce qui est utile quand l'on souhaite vendre ses scripts sans pour autant dévoiler ses connaissances).
- Zend Accelerator permet d'accélérer l'exécution des scripts PHP avec un système de cache (le site officiel indique des gains de rapidité jusqu'à 300 %).
- Zend IDE est un outil permettant de développer en PHP de manière plus conviviale.

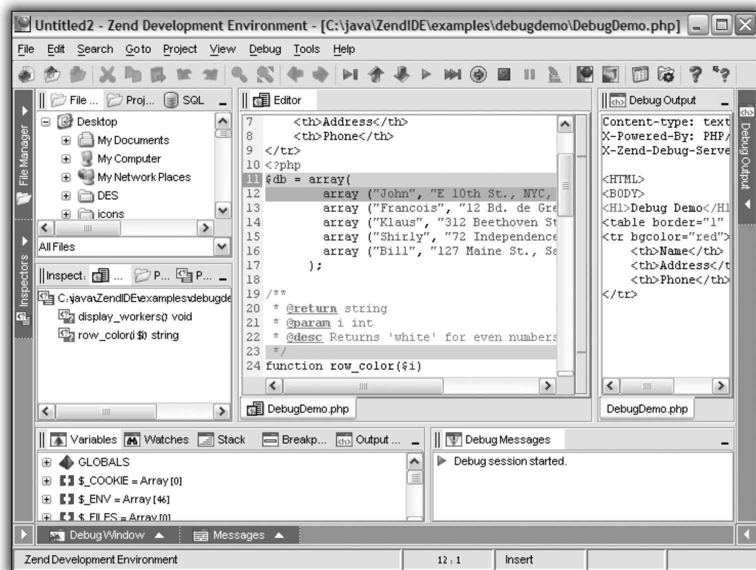


Figure 1.3 : *L'environnement de développement proposé par Zend*

La société Zend propose aussi un produit non open source mais gratuit : Zend Optimiser. Il s'agit d'une librairie qui permet d'optimiser à la volée votre code au moment où celui-ci est exécuté.



Argent et open source

Les dollars et l'open source n'ont jamais fait bon ménage. La réussite de cette société est donc d'autant plus remarquable. En vendant des solutions à des sociétés (qui en ont souvent largement les moyens), Zend est en mesure d'employer des développeurs qui travaillent 100 % de leur temps sur PHP. Toute la communauté PHP profite ainsi directement du succès et de la bonne santé financière de la société Zend.

Le rouleau compresseur PHP est désormais en marche et nul ne sait où il s'arrêtera. Des conventions et des conférences ont lieu désormais tous les mois autour de ce langage ; des milliers de sites, des centaines d'ouvrages, et même des magazines vendus en kiosques se consacrent maintenant exclusivement à PHP. Même le gouvernement français s'intéresse au phénomène et donne des instructions auprès de ses ministères afin que leurs sites soient conçus sur des solutions Lamp (par exemple SPIP Agora). Le Web est devenu central, aussi bien dans notre vie de tous les jours que dans la vie des entreprises, et PHP est en passe de devenir l'un des vecteurs déterminants de son expansion.



Technologies Lamp

Toute personne s'intéressant au PHP a dû croiser dans la littérature le terme *Lamp*. Ce sigle signifie Linux/Apache/MySQL/PHP. Lamp est un environnement complet permettant de faire fonctionner une application web. Il dispose d'un système d'exploitation (Linux), d'un serveur web (Apache), d'un système de gestion de bases de données (MySQL) et d'un langage de programmation (PHP). Il s'agit aujourd'hui, et de loin, de l'environnement le plus performant, le plus sûr et le plus abordable du marché.

Finissons ce paragraphe avec deux chiffres qui devraient marquer les esprits : plus de 1 million de serveurs web et presque 20 millions de sites exploitent aujourd'hui PHP.

1.3. Internet, comment ça marche ?

Dans cette partie, nous allons essayer de comprendre comment fonctionnent le Web et Internet en général. Une bonne compréhension de cette couche réseau vous permettra de mieux envisager le fonctionnement et l'importance de PHP.

Web et autres protocoles

Le Web est un réseau mondial de machines parlant la même langue. En informatique, cette langue est appelée « un protocole ». Le protocole du Web est l'HTTP (Hypertext Transfer Protocol).

Le Web n'est qu'un réseau parmi tant d'autres, HTTP a en effet de nombreux « cousins ».

Tableau 1.2 : *Quelques protocoles de haut niveau et leur fonction*

Protocole	Signification	Fonction
FTP	File Transfer Protocol	Transfert de fichiers
IRC	Internet Relay Chat	Dialogue en direct
NNTP	Network News Transfer Protocol	Envoi, lecture de news
POP	Post Office Protocol	Récupération des courriels
SMTP	Simple Mail Transfer Protocol	Envoi des courriels

Chacun a donc un rôle qui lui est propre et de nouveaux réseaux se créent tous les jours pour répondre aux nouveaux besoins des internautes. Nous avons ainsi vu depuis quelques années l'émergence des réseaux P2P de type BitTorrent qui permettent la recherche et l'échange de fichiers.

Créer un nouveau protocole ne consiste en fait qu'à définir une nouvelle langue compréhensible à la fois par un client et par un serveur.

Tableau 1.3 : *Exemple simplifié d'une définition de protocole*

Requête client	Réponse serveur
HELLO	HELLO

Tableau 1.3 : *Exemple simplifié d'une définition de protocole*	
Requête client	**Réponse serveur**
RECEVOIR *toto.txt*	Envoyer le fichier *toto.txt*
MESSAGE PAUL bonjour	Envoyer le message « bonjour » à l'utilisateur PAUL

En réalité, l'acceptation d'un nouveau protocole en tant que standard est extrêmement compliquée. Tout doit être parfaitement « ficelé ». Les cas les plus bizarres doivent avoir été considérés. La page **http://sunsite.dk/RFC/rfc/rfc2616.html** présente le protocole HTTP version 1.1 dans toute sa richesse et sa complexité.

Chaque protocole nécessite une application cliente qui lui est propre. Pour le HTTP, c'est un navigateur web ; pour le NNTP, c'est un lecteur de news ; pour le FTP, il s'agit d'un client FTP.

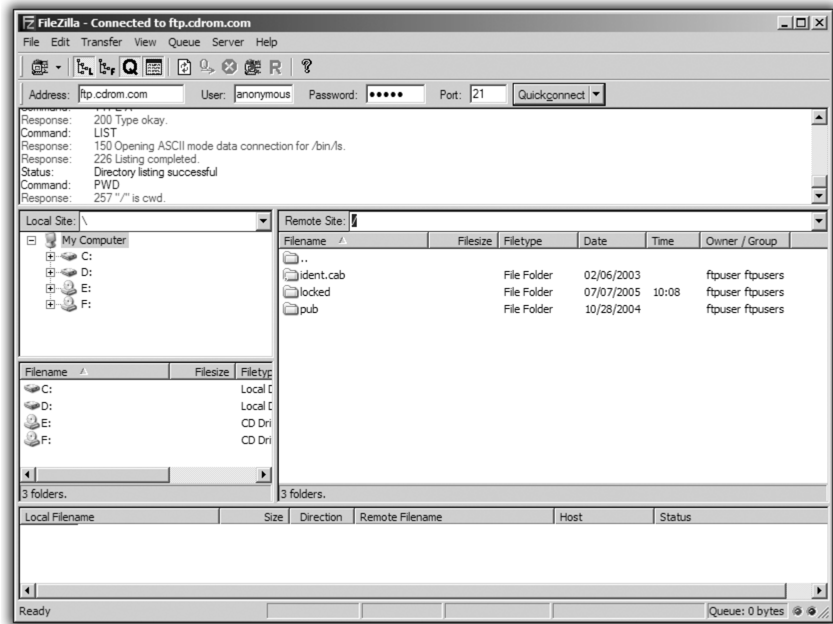


Figure 1.4 : *Client FTP listant les fichiers disponibles à l'adresse ftp://ftp.cdrom.com*

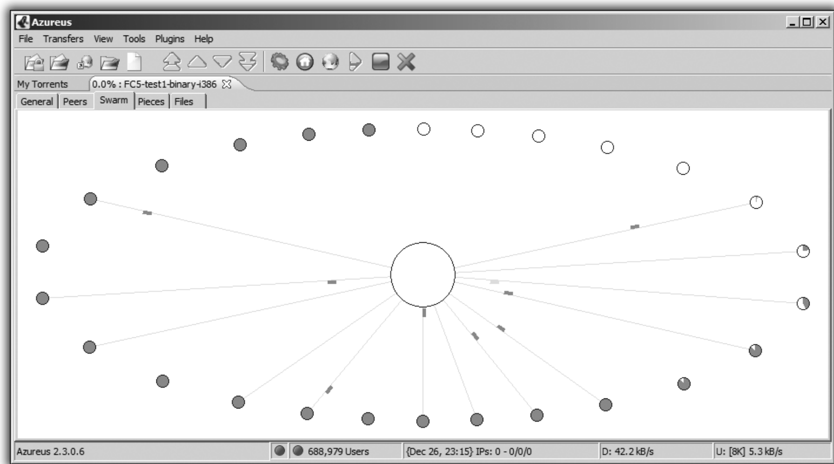


Figure 1.5 : Azureus, logiciel permettant de télécharger sur le réseau BitTorrent

Cependant, de plus en plus d'applicatifs permettent d'avoir accès à plusieurs protocoles. Ainsi, le logiciel Outlook Express, de Microsoft, permet de lire des courriels (POP) et des news (NNTP). Les navigateurs, quant à eux, permettent souvent d'avoir accès au protocole FTP. Si nous voulons avoir accès au serveur FTP ayant pour adresse **ftp.cdrom.com**, il suffit de taper l'URL (Uniform Resource Locator) `ftp://ftp.cdrom.com`.



Figure 1.6 : Un navigateur web accédant au serveur FTP ftp.cdrom.com

TCP/IP et Internet

Malgré cette profusion de protocoles, une chose ne change pas ; ces réseaux sont tous fondés sur un protocole sous-jacent unique : TCP/IP (Transmission Control Protocol/Internet Protocol). C'est cette combinaison de réseaux basés sur TCP/IP que l'on appelle Internet. Le réseau Internet est vraiment l'élément fondateur qui a permis l'émergence du Web, du mail et de tous ces services dont nous ne pourrions plus nous passer aujourd'hui. Comme pour beaucoup d'avancées scientifiques, cette invention est d'origine militaire. Dans les années 1960, une équipe de chercheurs plancha sur un système permettant d'assurer la continuité des échanges d'informations sensibles (entre des postes stratégiques), et cela même si certains postes (et donc certaines liaisons) étaient détruits. De cet impératif naquit TCP/IP. Ce protocole permet de découper les informations (pages web, courriels, images) en petits paquets et de les acheminer (de les « router ») d'un point à un autre. L'idée originale est la suivante : pour arriver à la même destination, tous ces paquets ne sont pas obligés de passer par la même route. Il est ainsi possible qu'un paquet passe par l'Asie pour aller de la France vers l'Angleterre si aucune autre route n'est à ce moment disponible. Il faut donc imaginer Internet comme un maillage mondial de serveurs interconnectés. C'est du fait de cette architecture que l'on parle, à propos du Web, de « toile d'araignée ».

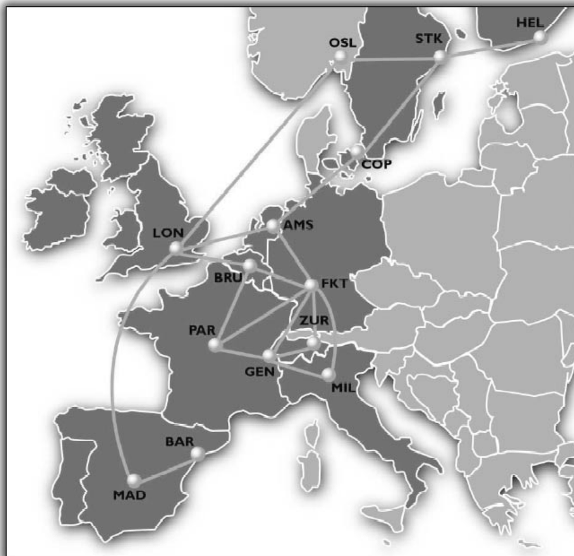


Figure 1.7 : *Quelques interconnexions européennes de la société Cable and Wireless*

Et le fournisseur d'accès dans tout ça ?

Qu'il s'agisse de Free, de Wanadoo, d'AOL, etc., son rôle est de relier votre ordinateur aux autres machines présentes sur Internet. La ligne téléphonique assume alors la fonction de lien. Votre modem sert à faire transiter les données informatiques entre Internet et votre ordinateur. Comme les modems classiques sont très lents, d'autres méthodes sont maintenant proposées pour créer un canal entre Internet et vous : le câble (le même qui vous permet d'accéder aux chaînes de télévision), les lignes téléphoniques « boostées » (l'ADSL), le satellite (pour les plus fortunés), les ondes radio et bientôt vos prises électriques, le WiMax.

Le serveur web

Le Web est donc un protocole applicatif fonctionnant sur un mode client-serveur.

Quand l'internaute souhaite voir la page *information* du site monsite.com, située à l'adresse (URL) **www.monsite.com/info.html**, il utilise un navigateur de type Internet Explorer, Netscape, Firefox, Opera, Lynx, Konqueror…

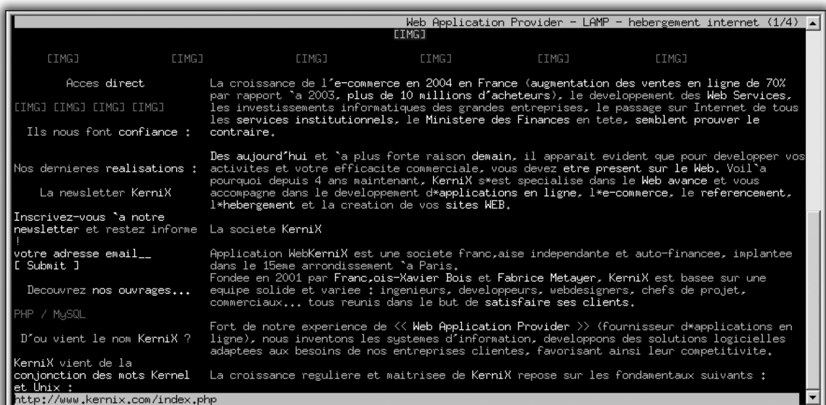


Figure 1.8 : *Le site kernix.com vu depuis le navigateur ELinks (mode texte) sous Unix*

Le navigateur va ensuite demander au serveur ayant l'adresse **www.monsite.com** de lui transmettre la page ayant pour nom *info.html*.

Essayons de comprendre plus en détail comment cet échange se déroule.

Étape 1 : le navigateur envoie une requête

Vous commencez par écrire l'URL `http://www.google.fr/index.html` dans votre navigateur.



Figure 1.9 : *Affichage d'une URL dans un navigateur*

Une fois cette adresse validée, le navigateur va enchaîner différentes actions.

Il commence par regarder quel type de protocole va être utilisé. Si l'URL commence par `http://`, c'est le protocole HTTP. En revanche, s'il s'agit de `ftp://`, le navigateur devra alors utiliser le protocole FTP.

Il doit ensuite découvrir où se trouve le serveur web **www.google.fr**. Pour cela, il va utiliser un autre service du Net : les DNS (*Domain Name Server* ; des adresses DNS sont systématiquement fournies par votre fournisseur d'accès). Les DNS sont des serveurs qui permettent d'associer un nom de domaine (**www.google.fr**) à une IP (`216.239.39.101`), un peu comme les pages blanches associent M. Dupont à son numéro de téléphone 01 02 03 04 05. Cette adresse IP est unique et identifie par conséquent de manière tout aussi unique le serveur web. C'est donc grâce à cette adresse IP que votre navigateur va pouvoir repérer le bon serveur sur Internet et entrer en contact avec lui. Comme il s'agit, dans ce cas, du protocole HTTP, le navigateur va communiquer avec le serveur HTTP (web) du serveur `216.239.39.101`. La requête qui sera faite vise à obtenir la page ayant pour nom *index.html*.



IP de Google

Si vous tapez l'URL `http://216.239.39.101`, vous arrivez sur la même page, ce qui est d'ailleurs plutôt rassurant.

Étape 2 : le serveur web retourne le fichier

Le serveur web a donc reçu une requête d'une machine qui souhaite obtenir le fichier *index.html*. Il va donc chercher sur son disque dur le fichier *index.html*, récupérer son contenu, puis envoyer ce flux de données au navigateur. Il sait où le renvoyer car, dès que vous êtes sur Internet, vous disposez vous aussi d'une adresse IP visible de l'extérieur.



Votre adresse IP

Si vous êtes relié à Internet et que votre système d'exploitation est Windows, vous pouvez connaître votre adresse IP en tapant la commande `IPCONFIG` (la commande peut être exécutée sous DOS ou directement depuis le menu **Démarrer/Exécuter**). Si, ensuite, vous installez un serveur HTTP (apache, IIS) ou FTP (WS_FTP Serveur) sur votre machine, vous serez en mesure d'être « vu » depuis l'extérieur. Si vous transmettez votre adresse IP à un ami, celui-ci pourra se connecter directement à votre machine avec un navigateur ou un client FTP. Votre machine joue alors le même rôle qu'un serveur d'hébergement, à la différence près que votre liaison n'est, dans tous les cas, pas très rapide. Sachez, par contre, que votre IP n'est pas fixe. À chaque fois que vous vous connectez ou déconnectez, vous changez d'IP.

```
Sélectionner Invite de commandes
Microsoft Windows XP [version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\fx>ipconfig

Configuration IP de Windows


Carte Ethernet Connexion au réseau local:

        Suffixe DNS propre à la connexion :
        Adresse IP. . . . . . . . . . . . : 192.168.0.203
        Masque de sous-réseau . . . . . . : 255.255.255.0
        Passerelle par défaut . . . . . . : 192.168.0.253

C:\Documents and Settings\fx>
```

Figure 1.10 : *L'adresse IP est ici 192.168.0.203*

En revanche, si vous aviez demandé l'URL **www.google.fr/intl/fr/about.html**, le serveur serait rentré dans le répertoire *intl*, puis dans le répertoire *fr*, et il aurait trouvé à cet endroit le fichier *about.html*.

Vous comprenez donc qu'une URL est composée de plusieurs éléments :

- le protocole **http://** ;
- l'adresse Internet du serveur, **www.google.fr** (adresse qui peut donc être une adresse IP) ;
- le chemin du fichier `/intl/fr/about.html`.

Le chemin est similaire à un chemin sur votre disque dur (par exemple *C:\tmp\message.txt*), sauf qu'il est noté sous la norme des chemins Unix : */tmp/message.txt*, où la première barre oblique (/) correspond à la racine.



Majuscules ou minuscules dans les URL ?

Pour répondre à cette question, il est nécessaire de diviser le problème en deux. Le nom de domaine (par exemple www.google.fr) d'un côté et le chemin d'accès au fichier (par exemple */index.html*) de l'autre. Les DNS n'étant pas sensibles à la casse (à la différence majuscule/minuscule), les URL http://google.fr et http://wWW.gOOglE.fR sont donc équivalentes.

En ce qui concerne les chemins, le problème est plus compliqué. Si le serveur HTTP fonctionne sous Unix, il sera sensible à la casse. Les chemins */index.html* et */index.htmL* seront donc différents. S'il fonctionne, par contre, sous Windows, le serveur ne sera pas sensible à cette différence. Les serveurs HTTP fonctionnant essentiellement sous Unix/Linux, il est donc préférable de faire attention à la façon d'écrire le chemin d'accès à une page ou à un script.

Étape 3 : le navigateur traite le fichier

Le navigateur reçoit donc le contenu du fichier *index.html*. Comme l'extension du fichier est *.html*, il sera traité comme un fichier HTML (voir Figure 1.11).

Le navigateur ne reçoit dans un premier temps que le contenu textuel de la page. Il doit donc, avant d'afficher la page, récupérer toutes les images contenues dans celle-ci. Il regarde dans le code, trouve toutes les adresses des images et fait des requêtes au serveur web pour les obtenir une par une. Bien évidemment, ces sous-requêtes sont totalement transparentes pour vous. Vous êtes cependant en mesure de les réaliser vous-même : quand le navigateur a besoin de l'image de titre, la requête est la suivante : http://www.google.fr/images/title_homepage4.gif. Si vous tapez cette URL dans votre navigateur, vous n'obtenez alors, comme prévu, que l'image de titre (voir Figure 1.12).

```
<html><head><meta http-equiv="content-type" content="text/html;
charset=UTF-8"><title>Google</title><style><!--
body,td,a,p,.h{font-family:arial,sans-serif;}
.h{font-size: 20px;}
.q{color:#0000cc;}
//-->
</style>
<script>
<!--
function sf(){document.f.q.focus();}
function
rwt(el,ct,cd,sg){el.href="/url?sa=t&ct="+escape(ct)+"&cd="+escape(cd)+"&url="+escap
e(el.href).replace(/\+/g,"%2B")+"&ei=2fShQ9CvFZOgiAKKlKGdDA"+sg;el.onmousedown="";r
eturn true;}
// -->
</script></head><body onload="sf()" topmargin="3" alink="#ff0000" bgcolor="#ffffff"
link="#0000cc" marginheight="3" text="#000000" vlink="#551a8b"><center><table
border="0" cellpadding="0" cellspacing="0" width="100%"><tbody><tr><td
align="right" nowrap="nowrap"><font size="-1"><a
href="https://www.google.com/accounts/Login?continue=http://www.google.fr/&amp;hl=e
n">Sign in</a></font></td></tr><tr height="4"><td><img alt="" height="1"
width="1"></td></tr></tbody></table><table border="0" cellpadding="0"
cellspacing="0"><tbody><tr><td align="right" valign="bottom"><img
src="Google_fichiers/hp0.gif" alt="Google" height="78" width="158"></td><td
valign="bottom"><img src="Google_fichiers/hp1.gif" alt="" height="78"
width="50"></td><td valign="bottom"><img src="Google_fichiers/hp2.gif" alt=""
height="78" width="68"></td></tr><tr><td class="h" align="right"
valign="top"><b></b></td><td valign="top"><img src="Google_fichiers/hp3.gif" alt=""
height="32" width="50"></td><td class="h" valign="top"><font style="font-size:
16px;" color="#6f6f6f"><b>France</b></font></td></tr></tbody></table><br>
<form action="/search" name="f"><script><!--
function qs(el) {if (window.RegExp && window.encodeURIComponent) {var
ue=el.href;var
qe=encodeURIComponent(document.f.q.value);if(ue.indexOf("q=")!=-1){el.href=ue.repla
```

Figure 1.11 : *Voici le contenu du fichier index.html que reçoit le navigateur*



Figure 1.12 : *Requête permettant de n'obtenir que l'image de titre*

Une fois tous les éléments constitutifs de la page récupérés par le navigateur, celle-ci peut être affichée.

PHP

Vous avez vu que lorsque le navigateur fait une requête sur un fichier HTML ou sur une image le serveur lui retourne le contenu du fichier tel quel, sans lui apporter la moindre modification.

Quand, par contre, la requête est faite sur un fichier disposant d'une extension *.php*, tout se passe différemment. Si le serveur web est compatible PHP, celui-ci va traiter tous les fichiers *.php* comme des scripts et transmettra par conséquent le flux de données (le contenu du fichier) à l'interpréteur PHP avant de l'envoyer à l'internaute. Cet interpréteur aura donc pour charge d'évaluer (d'interpréter) le code source PHP et de remplacer les lignes de code par leurs résultats.

Prenons l'exemple d'un fichier *test.php* contenant le code suivant :

```
<?php
  print("bonjour monde");
?>
```

Avant d'être transmis à l'interpréteur, le flux de données contient encore le code ci-dessus. Une fois le travail de l'interpréteur terminé, le flux interprété ne contient plus que `"bonjour monde"`. C'est précisément cette phrase `"bonjour monde"` qui sera transmise et qui apparaîtra dans votre navigateur.

Au niveau du serveur web, PHP tel un « filtre » modifie donc à la volée le flux de données.

Il devient évident qu'un script écrit en PHP qui n'aurait pas d'extension serait considéré comme un simple fichier texte (ou HTML) et ne serait pas traité en tant que script PHP (son code ne serait pas interprété).

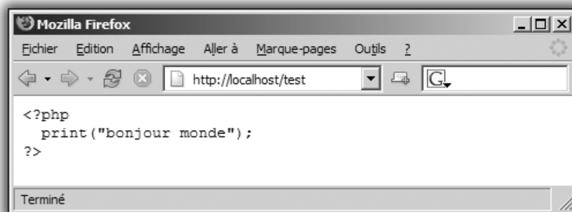


Figure 1.13 : *Sans extension, le script est considéré comme du simple texte*

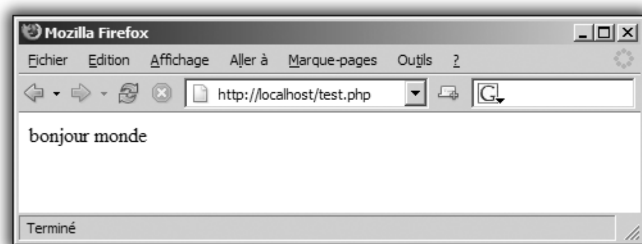


Figure 1.14 : *Avec une extension .php, le script est reconnu comme un script PHP ; il est donc interprété*

Le script doit avoir une extension particulière pour être reconnu en tant que script PHP. L'extension la plus répandue est *.php* (par exemple *test.php*). Il est néanmoins possible de rencontrer d'autres extensions : *.php3* ou *.phtml*. L'hébergeur précise généralement quelle extension doit être utilisée. Si le choix vous est offert, utiliser l'extension *.php* semble plus logique car nous en sommes aujourd'hui à la version 5 de PHP. De plus, cette extension risque fort de devenir la norme.

Les autres langages du Web

Les personnes intéressées par les technologies liées au Web auront pu s'apercevoir de la grande quantité de langages qui fourmillent sur la Toile.

Nous nous sommes déjà arrêtés sur les langages interprétés au niveau serveur tels que PHP, Perl, CFM, ou ASPX. Il existe une autre catégorie de programmes qui, eux, sont exécutés au niveau du client (dans le navigateur) : les Javascripts, les applets Java et les animations Flash.

Javascript

Expliquons rapidement la différence de fonctionnement entre ces deux modes.

- Niveau serveur : le script est exécuté sur le serveur à la suite d'une requête d'un navigateur. Le client reçoit ainsi une page prête à être affichée.
- Niveau client : la page retournée par le serveur web contient du code. C'est au niveau du navigateur que ce code va être exécuté.

Étudions deux exemples :

Listing 1-8 : test.php

```
<html>
<body>
<?php
print("test");
?>
</body>
</html>
```

Listing 1-9 : test.html

```
<html>
<body>
<script language=javascript>
document.write("test");
</script>
</body>
</html>
```

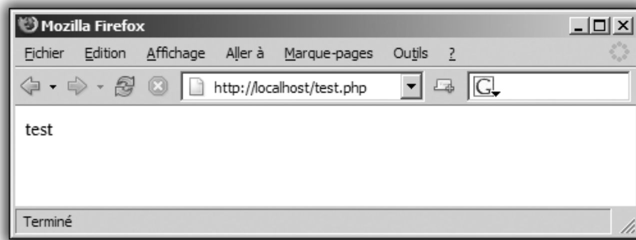


Figure 1.15 : *Résultat obtenu avec test.php ou test.html*

Ces deux programmes donnent un résultat identique, mais sont fondamentalement différents.

Dans le premier cas, le navigateur reçoit une page qui contient déjà le mot `test`, il peut donc l'afficher instantanément. Dans le deuxième cas, en revanche, il reçoit une page qui contient du code Javascript. Il doit par conséquent exécuter ce code avant d'afficher la page. Un navigateur ne gérant pas le Javascript n'aurait rien affiché à l'écran.

Chaque mode a bien évidemment ses inconvénients.

- Au niveau serveur : le fait de devoir interpréter le script dès qu'une requête lui parvient est très lourd à gérer pour le serveur et peut conduire à des ralentissements au niveau de la livraison des pages. Dans le cas de l'exemple ci-dessus, si 10 000 personnes demandent la page *test.php*, le serveur devra interpréter 10 000 fois le script. Si en revanche 10 000 personnes demandent

la page *test.html*, le serveur se contente d'envoyer 10 000 fois la page et les 10 000 exécutions se feront chez les clients. Ce mode permet de répartir la charge de travail et de n'avoir aucun ralentissement. Dans le cas ci-dessus, les deux versions se valent, mais imaginez un script censé chercher et afficher la 100 000e décimale du chiffre pi !

- Au niveau client : le Javascript est un langage certes normalisé, mais qui est géré de façon plus ou moins performante, avec plus ou moins de fonctionnalités selon les navigateurs. Il est ainsi très difficile de rendre un applicatif Javascript exécutable sur tout type de système.



Incompatibilités entre navigateurs

Bien que des normes soient édictées régulièrement par le W3C pour faire évoluer le Web, nous pouvons hélas ! déplorer le fait que les différents navigateurs ne les suivent pas scrupuleusement. Microsoft notamment, avec Internet Explorer, a particulièrement compliqué la vie des développeurs web en leur imposant d'écrire des codes sortant de la norme afin de rester compatible avec son navigateur vedette.
L'émergence de Firefox et plus généralement du monde du libre (free software) commence néanmoins à marquer les esprits et Microsoft devrait avec son futur Internet Explorer 7 gagner en compatibilité (Javascript, CSS, DOM).

Java

Il est important, avant de clore ce chapitre, de s'intéresser quelques instants à Java. Développé par la société Sun Microsystem, initialement pour des composants embarqués, ce langage a fait l'effet d'une bombe lors de sa sortie. Le principe était relativement simple et révolutionnaire : créer un langage qui permette de tirer profit des avantages du langage interprété et du langage compilé. Au lieu de compiler des sources Java directement en un binaire propre à un type de processeur, la compilation se fait dans un langage intermédiaire (*bytecode*), qui doit être interprété, par la suite, par une machine virtuelle. L'interprétation est alors facilitée et correspond plus à une conversion.

Il fut très vite évident que Java aurait son rôle à jouer sur le Web, où foisonnaient une multitude de systèmes (Mac, PC, Windows, Unix, etc.), et ce fut par l'intermédiaire des applets qu'il se fit connaître du grand public. Une applet java est un programme Java qui s'exécute dans un navigateur. À la différence du Javascript, il est possible, en Java, de

construire des applications disposant d'interface graphique complexe. Des applets de tableurs, de dessins, de chats firent ainsi leur apparition sur le Web à la stupéfaction générale.

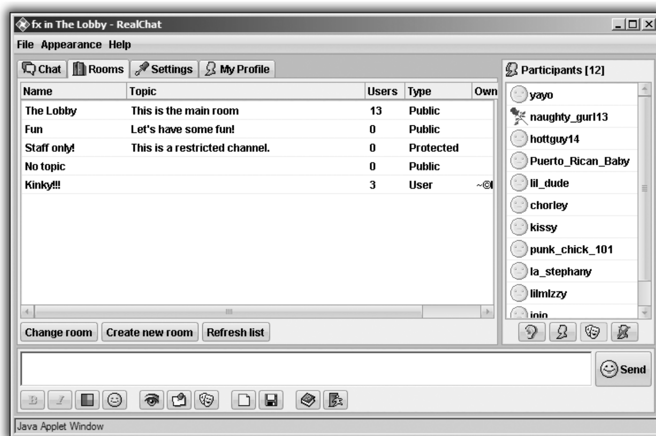


__Figure 1.16 :__ Un logiciel de chat dans un navigateur web

À la plus grande joie des internautes, les applets Java permirent aussi l'arrivée des jeux sur le Web.

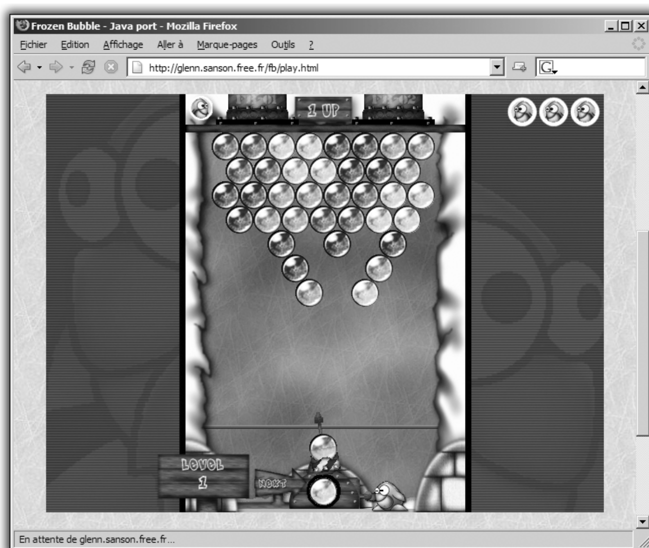


__Figure 1.17 :__ Spaceball : jeu écrit en Java, fonctionnant sur tout type de plateforme

Hélas, les applets sont des programmes qui demandent beaucoup de ressources machine (processeur, mémoire) ! Elles nécessitent en outre la présence d'une machine virtuelle Java et sont généralement lentes à charger. Toute la politique de Sun fut donc de transférer la technologie Java vers le niveau serveur avec les servlets. Avec certaines extensions, le serveur Apache (via son cousin Tomcat) est désormais en mesure d'interpréter du Java aussi facilement que du code PHP.

Animation Flash

Le Flash est un format développé par la société Macromedia qui permet de créer des animations sur le Web. Alors qu'il s'agissait au départ d'un outil essentiellement graphique destiné aux designers web, Macromedia a vite compris qu'il disposait d'une véritable bombe et qu'il pouvait en faire un véritable environnement de développement pour le Web. L'environnement Flash MX permet désormais de réaliser des interfaces graphiques complètes, d'interagir avec des services web, de manipuler les fichiers XML et de développer des applicatifs en utilisant le langage interne aussi puissant que complet qu'est ActionScript.



FLA et SWF

Deux formats de fichiers sont liés au Flash : le FLA qui peut être comparé au fichier source et le SWF qui correspond à l'exécutable. Si vous souhaitez apporter des modifications, vous devez donc disposer du FLA pour l'ouvrir dans Flash MX. Au contraire, pour exécuter l'animation au sein d'un navigateur, le SWF vous sera nécessaire.

Flash a l'avantage d'avoir un rendu identique sur tous les navigateurs à la condition, certes très restrictive, que l'ordinateur dispose du plug-in Flash. Lorsque l'on sait que Macromedia est un concurrent de Microsoft (éditeur de l'incontournable Windows) et que les distributions Linux rechignent à installer des logiciels non libres, nous ne sommes pas près de disposer du plug-in Flash préinstallé sur nos machines. Le caractère propriétaire de Flash à l'heure où les formats de fichiers tendent tous à s'ouvrir risque également de freiner l'adoption de cette technologie parmi les sociétés éditrices de logiciels en ligne.

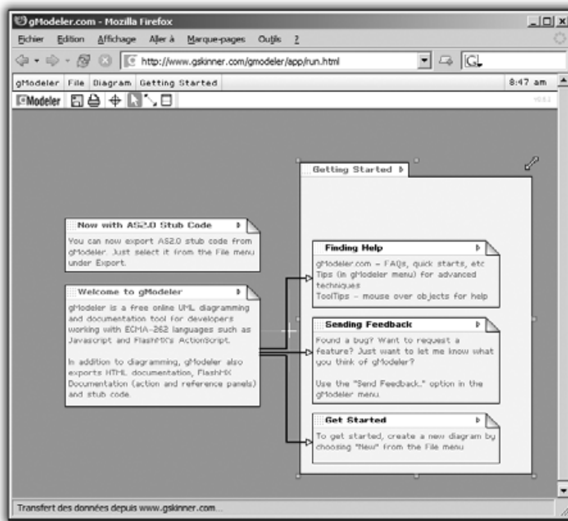


Figure 1.18 : *Outil de création de diagrammes intégralement réalisée en Flash*

1.4. Check-list

- PHP n'est qu'un langage de programmation parmi d'autre.
- PHP fait partie des langages interprétés.
- L'interprétation des scripts PHP est réalisée le plus souvent au niveau du serveur web dont le meilleur représentant est Apache.
- PHP est particulièrement adapté aux développements web.
- PHP fonctionne sur une multitude de plateformes.
- PHP est libre et gratuit.
- Le Web est une des dimensions d'Internet au même titre que les courriels ou le P2P.

Chapitre 2

L'environnement de travail

WampServer 48
Paramétrage de PHP 60
Check-list 64

L'objectif est ici de mettre en place sur votre machine un environnement de travail permettant de tester les exemples présentés dans la suite de l'ouvrage. À l'issue de ce chapitre, vous disposerez d'un serveur web capable d'interpréter des scripts PHP, d'un système de gestion de bases de données (MySQL) et d'un outil permettant d'interagir avec ce dernier : phpMyAdmin.

2.1. WampServer

Bien qu'Apache, MySQL, et PHP puissent être installés séparément sous Windows, le choix se portera ici sur un outil capable d'automatiser l'intégralité de ce processus.

Installation

Wamp Server peut être téléchargé sur le site **www.wampserver.com** sous la rubrique *Downloads*.

1 Double-cliquez sur l'archive que vous venez de télécharger

De nombreuses modifications ont été apportées à la version 2 de Wamp Server. Les créateurs conseillent par conséquent aux personnes disposant déjà d'une version de Wamp Server sur leur machine, de sauvegarder leurs développements, de désinstaller.l'ancienne version et enfin d'installer la toute nouvelle.



Compatibilité du code

Les développements (PHP, MySQL) réalisés sur une version antérieure de Wamp Server ont 99 chances sur 100 de fonctionner sur une version plus récente de Wamp Server. Les projets PHP et MySQL apportent en effet une importance énorme au fait de maintenir la compatibilité des développements d'une version à l'autre.

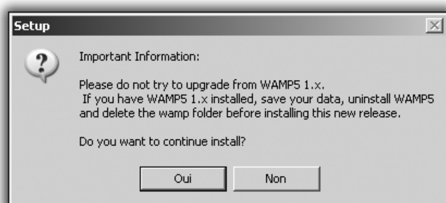


Figure 2.1 : *Avertissement*

2 La fenêtre d'installation se lance et vous présente votre version de Wamp.

Cette version est sans aucun rapport avec celle des outils qu'elle contient (que ce soit Apache, PHP ou MySQL).



Figure 2.2 : *l'installation de Wamp Server 2*

L'écran suivant permet d'accepter la licence.

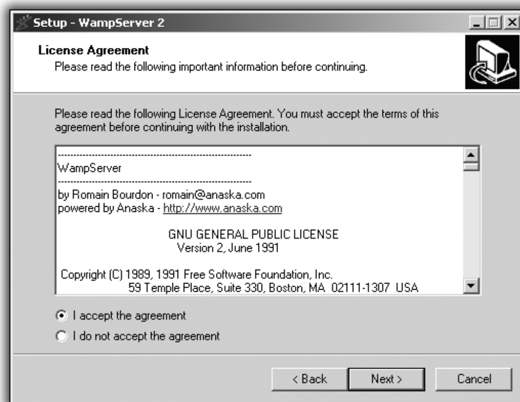


Figure 2.3 : *Acceptez...*

3 Précisez le répertoire qui contiendra l'ensemble des composants ainsi que vos sources.

Ce répertoire représentera approximativement 100 Mo de données. N’hésitez pas à sélectionner un autre disque si vous sentez que l’espace libre de votre partition *C:* est trop juste. Un changement d’emplacement dans un deuxième temps serait beaucoup plus compliqué.

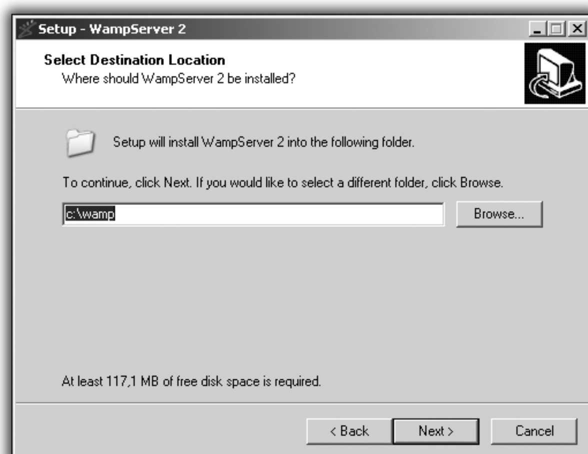


Figure 2.4 : *Indiquez le répertoire*

L’écran suivant permet d’obtenir des raccourcis sur le bureau pour lancer Wamp Server. Il n’est pas nécessaire de cocher ces options dans la mesure où il restera possible de démarrer Wamp Server en passant par le menu **Démarrer**.

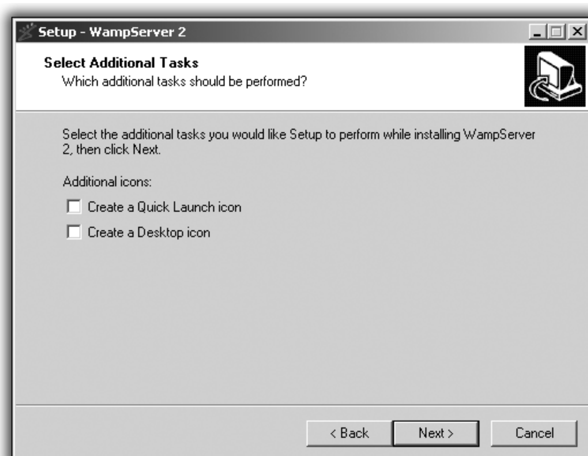


Figure 2.5 : *Raccourcis*

L'étape suivante résume vos différents choix d'installation.

4 Il est encore temps de les modifier en revenant en arrière par le bouton < **Back**.

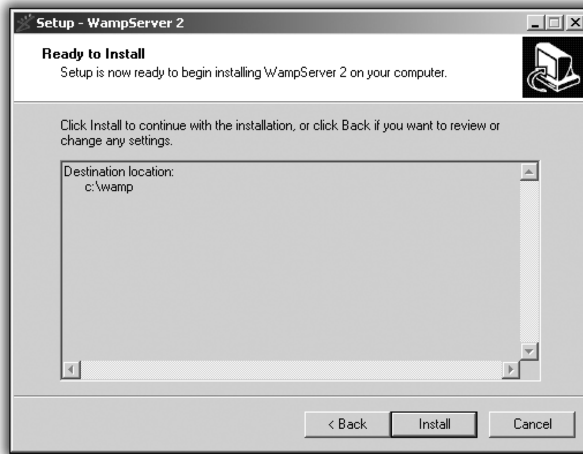


Figure 2.6 : *Résumé de l'installation*

La prochaine étape correspond à l'installation physique des composants sur votre machine. Pas loin de 2000 fichiers sont installés dans le répertoire *C:\wamp*.

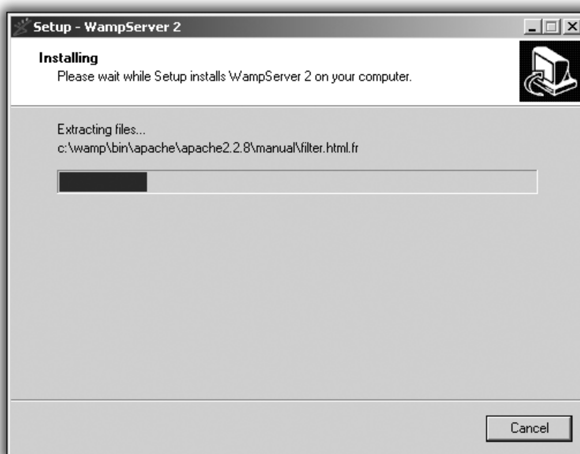


Figure 2.7 : *En cours…*

5 Définissez le navigateur qui sera utilisé pour ouvrir ces pages.

Nous vous conseillons d'utiliser le navigateur Firefox.

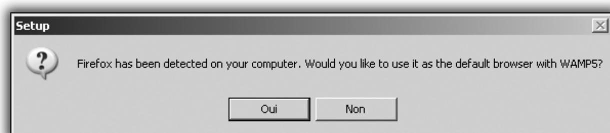


Figure 2.8 : *Emplacement de Firefox*

6 Indiquez votre adresse email ainsi que le nom de votre serveur d'envoi de mails. Ce serveur, dit SMTP, est différent pour chaque fournisseur d'accès à Internet. Il peut par exemple prendre les valeurs suivantes : `smtp.free.fr`, `smtp.wanadoo.fr`, `smtp.noos.fr`, `smtp.club-internet.fr`, etc.

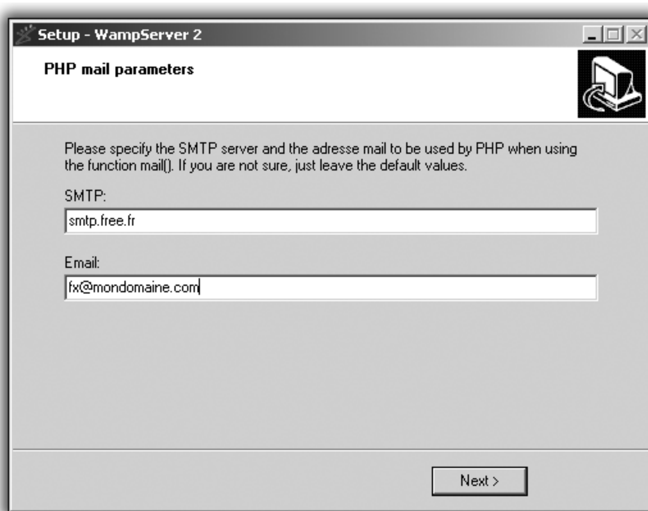


Figure 2.9 : *Serveur SMTP*

L'étape finale vous confirme que l'installation s'est bien déroulée et vous propose de démarrer sans plus attendre le Wamp Server.



Figure 2.10 : *Installation terminée*

Premiers pas

À l'issue de ces étapes, Apache, MySQL et PHP sont installés sur votre machine. Vous pouvez le vérifier en réalisant une première requête sur le domaine associé à votre machine : **http://localhost**.



Figure 2.11 : *tout fonctionne merveilleusement, Apache retourne la première page web*

La page affichée correspond au fichier *index.php* situé dans *C:\wamp\www*. Vous pouvez de la même manière faire appel à votre propre page en plaçant le fichier *test.html* dans ce même répertoire et en y faisant appel de la manière suivante : `http://localhost/test.html`.

Listing 2-1 : c:\wamp\www\test.html

```
<hr/>bonjour monde<hr/>
```

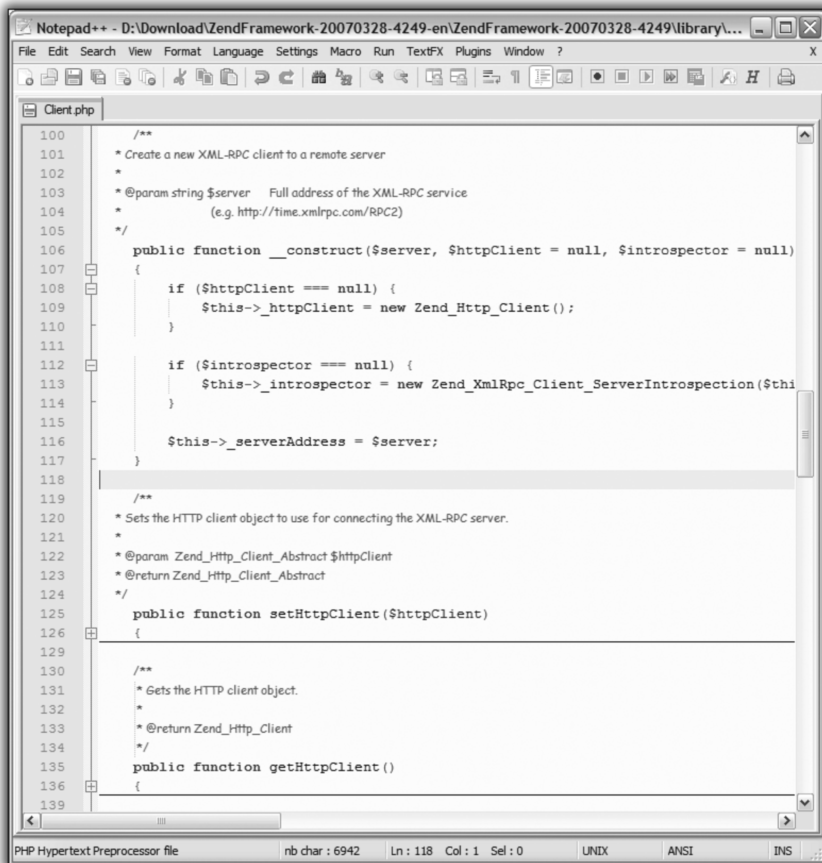


Figure 2.12 : *Les pages sont également accessibles*

Le démarrage de Wamp a enrichi votre System Tray d'une petite icône évoquant un compteur de vitesse.

Cette icône peut prendre différentes couleurs en fonction de l'état des serveurs.

- **Marron** : aucun service n'est démarré.
- **Jaune** : un seul service est démarré.
- **Blanche** : les deux services sont démarrés.



Figure 2.13 :
Wamp est démarré, tout est OK

Le gestionnaire de tâches de Windows confirme également que les programmes Apache (*httpd.exe*) et MySQL (*mysqld-net.exe*) résident bien en mémoire. La colonne *Util. mémoire* permet de quantifier la mémoire utilisée par ces services :

- *2 x Apache.exe* : 25 Mo.
- *1 x mysqld-nt.exe* : 12 Mo.
- *1 x wampmanager.exe* : 4 Mo.

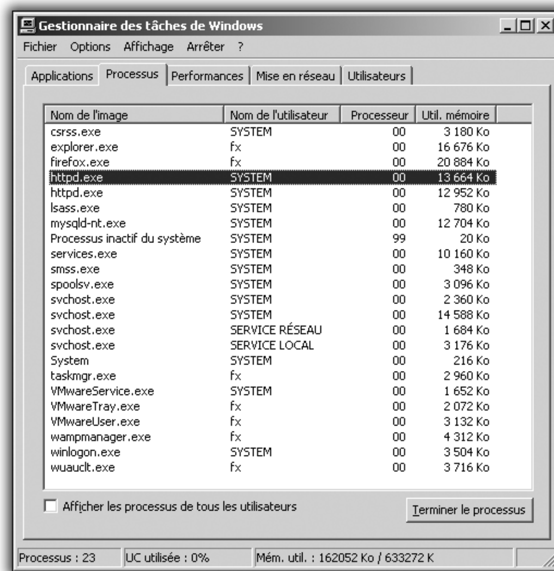


Figure 2.14 : *Gestionnaire de tâches Windows*

Si vous avez choisi de ne pas lancer Wamp Server automatiquement au démarrage, vous pouvez à tout moment le faire par le menu **Démarrer/Tous les programmes/WampServer/Start WampServer**.

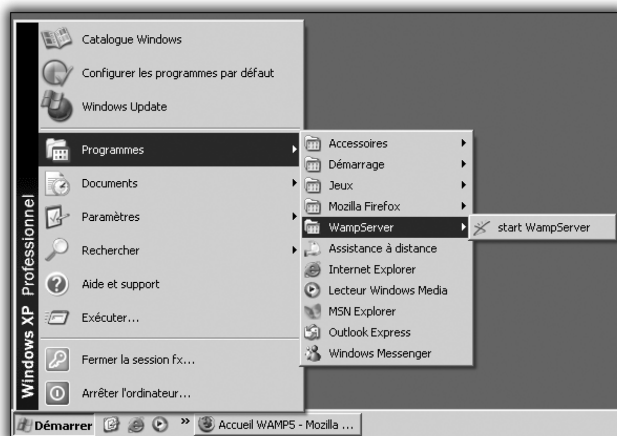


Figure 2.15 : *Menu d'accès*

Le menu de Wamp

En cliquant avec le bouton gauche de la souris sur l'icône de Wamp, vous affichez un petit menu déroulant.



Figure 2.16 : *Actions accessibles depuis le menu Wamp*

Ce menu permet :

- d'ouvrir directement des pages, notamment la page d'accueil et l'application phpMyAdmin ;

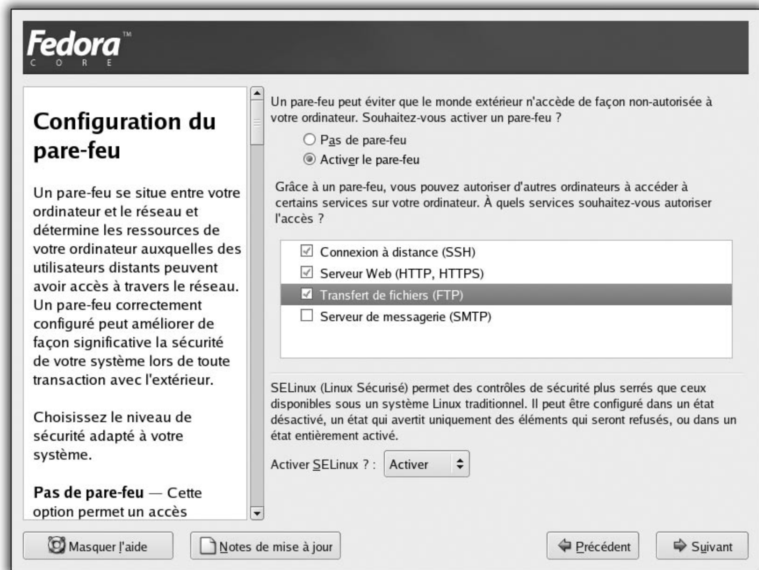


Figure 2.17 : *phpMyAdmin*

- d'ouvrir le répertoire *www* qui va contenir vos futurs scripts ;

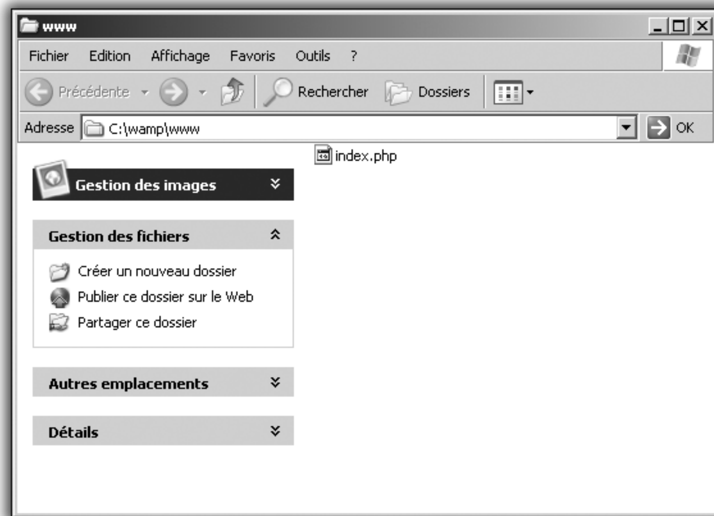


Figure 2.18 : *Le répertoire racine de du serveur web*

- de lire les fichiers LOG d'Apache, de MySQL et de PHP ;

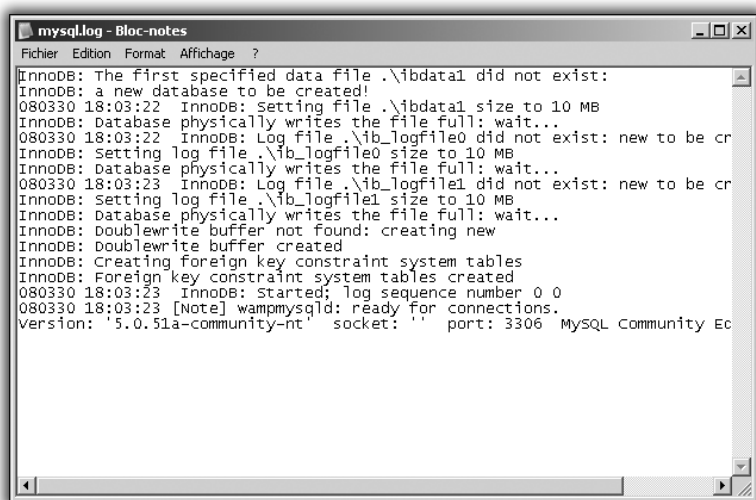


Figure 2.19 : *Un exemple de fichier LOG : mysql_error.log*

- d'éditer leur fichier de configuration ;

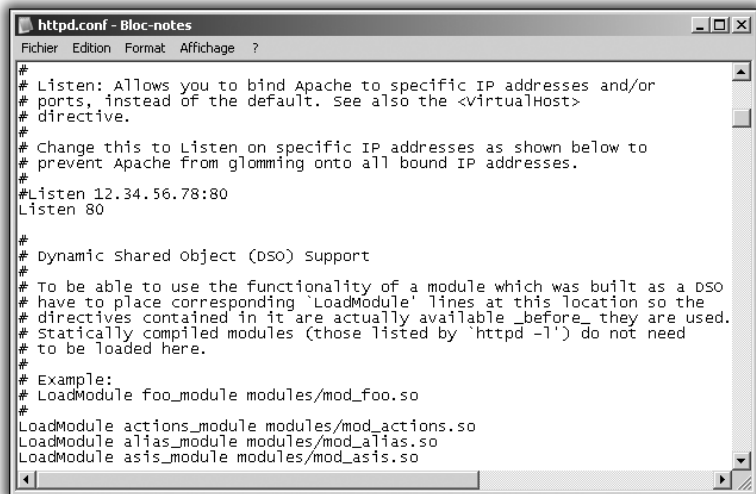


Figure 2.20 : *Un exemple de fichier de configuration : httpd.conf*

- de gérer les extensions PHP ;

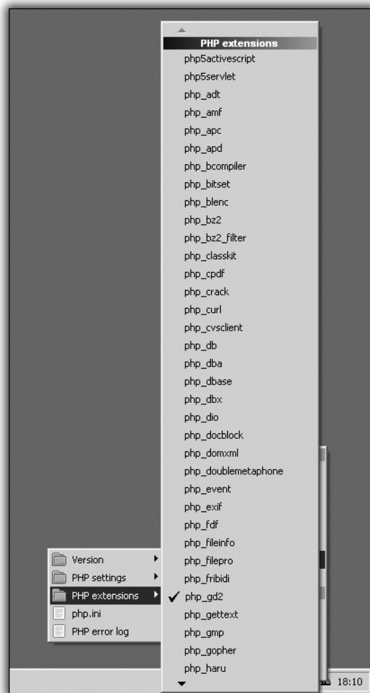


Figure 2.21 : *Chargement, suppression d'extensions PHP*

- d'arrêter, de démarrer et de redémarrer Apache et MySQL.



Figure 2.22 : *Menu d'Apache*

L'éditeur Notepad++

Les éditeurs de code PHP pullulent sur le Net. Un site web leur est même consacré : **www.php-editors.com/review**.

Chacun dispose bien évidemment de ses propres avantages et inconvénients. Notre choix se portera sur Notepad++ qui apparaît particulièrement adapté aux besoins du développeur PHP.

- Il est gratuit.
- Il est rapide.
- Il colorise le code.
- Il réduit les blocs de code (fonctions, class, structures de contrôle).



Figure 2.23 : *édition d'un script PHP avec Notepad++*

Cet éditeur peut être téléchargé à l'adresse **http://notepad-plus.sourceforge.net/fr/site.htm**.

2.2. Paramétrage de PHP

Au moment de sa mise en œuvre, PHP prend en compte un certain nombre de directives de configuration. Ces dernières, placées dans le fichier *php.ini*, permettent de paramétrer :

- le parseur PHP ;
- la sécurité ;

- les rapports d'erreur et les fichiers LOG ;
- la compatibilité avec les versions précédentes ;
- les extensions.

Sous Windows, ce fichier peut être ouvert via l'icône *Wamp Server* au sein du menu **PHP**.



Figure 2.24 : *L'accès aux fichiers de configuration*

Le fichier se situe physiquement dans le répertoire *C:\wamp\bin\apache\apacheX.X.X\bin*.

Tableau 2.1 : *Paramètres de configuration de PHP dans php.ini*

Paramètre	Valeur	Signification
`allow_url_fopen`	Booléen : `On` ou `Off`	Autorise l'ouverture de fichiers distants
`asp_tags`	Booléen	Autorise les balises ASP de type `<% %>`
`date.timezone`	String	Précise la zone géographique de la machine (par exemple `"Europe/Paris"`)
`default_mimetype`	String	Permet de préciser le type de fichier retourné par défaut par PHP (par exemple `"text/html"`)
`default_charset`	String	Permet de préciser l'encodage par défaut des caractères (par exemple `"iso-8859-1"`)
`display_errors`	Booléen	Autorise l'affichage des erreurs
`doc_root`	String	Définit le dossier racine

Tableau 2.1 : *Paramètres de configuration de PHP dans php.ini*

Paramètre	Valeur	Signification
`engine`	Booléen	Permet d'interdire l'usage de l'interpréteur PHP
`extension_dir`	String	Permet d'indiquer le répertoire contenant les extensions (par exemple `"c:/wamp/php/ext"`)
`error_log`	String	Définit le fichier dans lequel les erreurs seront « loggées »
`error_reporting`	Entier	Définit le niveau d'affichage des erreurs
`file_uploads`	Booléen	Autorise ou non le transfert (*upload*) de fichiers
`include_path`	String	Définit les répertoires dans lesquels les fonctions `require()` et `include()` vont chercher les fichiers (par exemple, sous Windows, `.;c:\wamp\www\boutique` ; sous Linux, `.:/var/web/html/boutique`)
`log_errors`	Booléen	Indique si les erreurs des scripts doivent être enregistrées dans le fichier LOG du serveur
`magic_quotes_gpc`	Booléen	Autorise l'échappement automatique des caractères `'`, `"`, `\` et `NUL`
`max_execution_time`	Entier	Définit le temps maximal d'exécution d'un script (en secondes)
`memory_limit`	Entier	Définit la mémoire maximale que peut utiliser un script (par exemple `8M`)
`open_basedir`	String	Restreint l'ouverture des fichiers à un répertoire spécifique
`register_globals`	Booléen	Indique si les variables EGPCS doivent être initialisées en tant que variables globales

Tableau 2.1 : *Paramètres de configuration de PHP dans php.ini*		
Paramètre	**Valeur**	**Signification**
`safe_mode`	Booléen	Permet de passer en mode « sécurisé » et de vérifier que le propriétaire d'un script est le même que celui du fichier accédé
`safe_mode_exec_dir`	String	Répertoire contenant les binaires pouvant être exécutés en mode sécurisé
`safe_mode_gid`	Booléen	Permet de vérifier que le groupe du propriétaire d'un script est le même que celui du fichier accédé
`session.save_path`	String	Répertoire de stockage des sessions
`session.name`	String	Nom du cookie de session (par exemple `PHPSESSID`)
`session.cookie_lifetime`	Entier	Durée de vie du cookie de la session ; la valeur `0` (par défaut) signifie que la session sera détruite avec la fermeture du navigateur
`short_open_tag`	Booléen	Autorise les balises raccourcies `<? ?>`
`upload_tmp_dir`	String	Précise le répertoire qui sera utilisé pour stocker temporairement les fichiers « uploadés »
`upload_max_filesize`	Entier	Précise la taille maximale des fichiers uploadés (par exemple `4M`)
`variables_order`	String	Définit l'ordre dans lequel PHP va initialiser ses variables globales ; la valeur `"EGPCS"` correspond à l'ordre suivant : `$_ENV`, `$_GET`, `$_POST`, `$_COOKIE`, `$_SESSION`
`zend.ze1_compatibility_mode`	Booléen	Permet à PHP 5 de se comporter comme PHP 4 au cœur du système PHP
`zlib.output_compression`	Booléen	Permet de compresser à la volée l'ensemble des données affichées par PHP

Toute modification apportée au fichier *php.ini* doit être accompagnée du redémarrage du serveur HTTP.



Le fichier httpd.conf

Ces valeurs peuvent également être précisées au sein du fichier de configuration d'Apache : *httpd.conf*. Il est même possible, en utilisant la directive `Virtualhost`, d'initialiser spécifiquement les différents sites présents sur la machine. Il pourrait ainsi être possible d'autoriser l'upload de fichiers pour un site, et de l'interdire pour un autre.

```
<VirtualHost *:80>

  ServerName       www.site.com
  DocumentRoot     /var/web/site

  php_admin_flag   engine              on
  php_admin_flag   file_uploads       off

</VirtualHost>
```

2.3. Check-list

- Wamp Server est un logiciel permettant d'installer très facilement sous Windows les logiciels Apache, MySQL et PHP.
- L'outil phpMyAdmin, lui-même écrit en PHP, permet de créer et de gérer les bases de données.
- Le fichier *php.ini* contient les directives de configuration de PHP.
- Une modification de ce fichier impose le redémarrage du serveur Apache.

Chapitre 3

Les fondamentaux

Structure d’un programme 67
Les commentaires 72
Les variables 74
Les constantes 78
Les types de données 80
Les structures de contrôle 86
Organisation du code 99
Check-list 113

Nous nous intéresserons dans ce chapitre à la syntaxe du PHP ainsi qu'aux différents éléments qui composent ce langage : les variables, les types de données, les fonctions et enfin les structures de contrôle. Ces éléments correspondent aux briques fondamentales qui permettront par la suite de réaliser des applications de plus grande envergure. Une bonne compréhension de ce chapitre est donc indispensable pour pouvoir passer sereinement aux chapitres suivants.

Avant d'entrer dans le vif du sujet, il est important de fixer un certain nombre de conventions. Vous avez lu précédemment que PHP est un langage de programmation interprété. En ce sens, un script écrit en PHP nécessite un autre composant pour être exécuté : l'interpréteur PHP. Par défaut, les systèmes d'exploitation les plus répandus (Windows, Mac OS) ne disposent pas de tels interpréteurs. Il existe donc deux possibilités pour faire fonctionner un programme écrit en PHP :

- utiliser l'environnement mis en place dans le chapitre précédent ;
- le placer chez un hébergeur prenant en charge PHP.

Nous allons considérer dans les prochains chapitres que vous travaillez sur votre propre machine. Comme nous l'avons vu dans le premier chapitre, toute machine dispose d'un nom par défaut : ici, *localhost*. Ce serveur aura donc pour adresse **http://localhost**. Ainsi, quand une mention sera faite d'un script *test1.php*, cela impliquera que vous aurez créé un fichier portant le nom *test1.php*, que vous l'aurez placé dans le répertoire prévu à cet effet et que vous aurez renseigné l'adresse dans votre navigateur **http://localhost/test1.php** pour l'exécuter. Une adresse de type **http://localhost/chap03/test2.php** signifierait quant à elle que vous avez créé un répertoire *chap03* dans le répertoire principal et que vous y avez placé le fichier *test2.php*.

Vous constaterez également le choix du navigateur web Firefox pour illustrer les exemples. En l'espace de quelques années, ce navigateur a pris jusqu'à 30 % de parts de marché à Internet Explorer. Ses avantages sont nombreux et variés :

- respect des standard (passés, présents et futurs) ;
- avancées technologiques (XUL, CANVAS, SVG) ;
- outils utiles aux développeurs web (affichage du code source, console Javascript) ;
- gratuité, sécurité, et rapidité ;

- mises à jour fréquentes et automatiques ;
- nombreuses extensions.



Firefox en français

Firefox peut être téléchargé en langue française sur le site www.mozilla-europe.org/fr/

3.1. Structure d'un programme

Un script écrit en PHP correspond à un fichier texte contenant des lignes de code (instructions). Vous savez depuis le premier chapitre que ce fichier texte doit avoir une extension de type *.php* pour pouvoir être évalué.

Les lignes de codes contenues dans un script PHP doivent être englobées entre les balises `<?php` et `?>` : elles forment alors un bloc de code.



Les balises `<%` et `%>`

Il est possible de rencontrer les balises d'ouverture et de fermeture `<%` et `%>`. Cette notation n'est cependant pas très répandue et devrait être abandonnée avec PHP6.

Écrivons, en suivant ce principe, notre premier programme PHP : *test.php*, un classique du genre.

Listing 3-1 : Votre premier programme PHP

```
<?php
print("bonjour monde");
?>
```

Il s'agit du programme le plus simple que l'on puisse imaginer. La première ligne, `<?php`, indique à l'interpréteur que les lignes qui vont suivre doivent être traitées comme du code PHP et qu'elles doivent donc être interprétées.

La deuxième ligne, `print("bonjour monde");`, est une instruction qui commande à l'interpréteur d'afficher à l'écran la phrase `"bonjour monde"`. En PHP, chaque instruction doit être ponctuée d'un point-virgule (`;`). Au sein du bloc de code, les instructions sont exécutées les unes après les autres.

L'expression `print()` correspond à ce que l'on appelle en programmation une « fonction ». Le rôle de cette fonction est d'afficher la donnée qui lui est adressée en paramètre.



Par défaut, PHP est livré avec un grand nombre de fonctions qui sont décrites pour la plupart dans le chapitre « Les fonctions PHP ».

Ce sont ces fonctions qui permettent, notamment, de travailler sur les nombres, les chaînes de caractères (mots, phrases), les tableaux, d'accéder aux bases de données, de générer des images, etc.

Enfin, la dernière ligne (`?>`) de ce script indique à l'interpréteur que les lignes qui suivent ne sont plus à interpréter. Elles peuvent donc être affichées telles quelles.

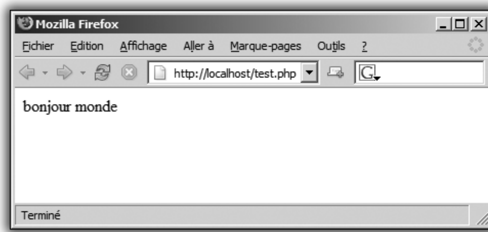


Figure 3.1 :
Le résultat de votre premier programme

Si vous voulez maintenant afficher deux messages, placez deux instructions dans le bloc de code :

Listing 3-2 : Votre script contient deux instructions

```
<?php
print("instruction 1");
print("instruction 2");
?>
```

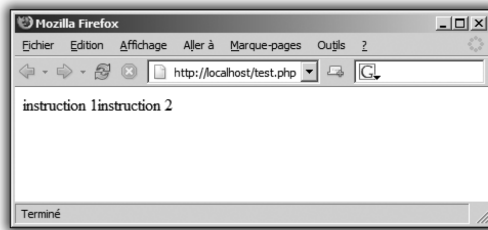


Figure 3.2 :
Résultat

Le résultat peut vous surprendre, les deux phrases se trouvant en effet sur la même ligne. En y réfléchissant bien, c’est cependant tout à fait normal. La première instruction affiche la phrase `"instruction 1"` et la deuxième affiche `"instruction 2"`. Votre navigateur va donc recevoir un fichier contenant `"instruction 1instruction 2"` et l’afficher tel quel. Si vous souhaitez avoir deux lignes différentes, rien de plus simple : ajoutez la balise HTML `<br/>` qui permet de réaliser un saut de ligne :

Listing 3-3 : Deux instructions

```
<?php
print("instruction 1<br/>");
print("instruction 2");
?>
```

La page que le navigateur reçoit contient alors la ligne `"instruction 1<br/>instruction 2"`, ce qui vous permet de séparer les deux lignes.

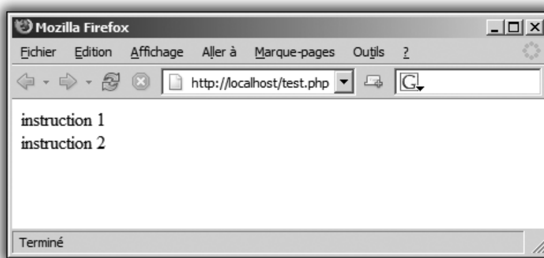


Figure 3.3 : *Le résultat correspond aux attentes*

L’expression `phpinfo()` est une autre fonction interne du PHP. Elle a pour rôle de donner des informations sur le PHP utilisé pour faire fonctionner votre script :

- version du PHP ;
- version des différents modules installés ;
- type de serveur web ;
- données système disponibles…

Listing 3-4 : Informations sur la version de PHP

```
<?php
phpinfo();
?>
```

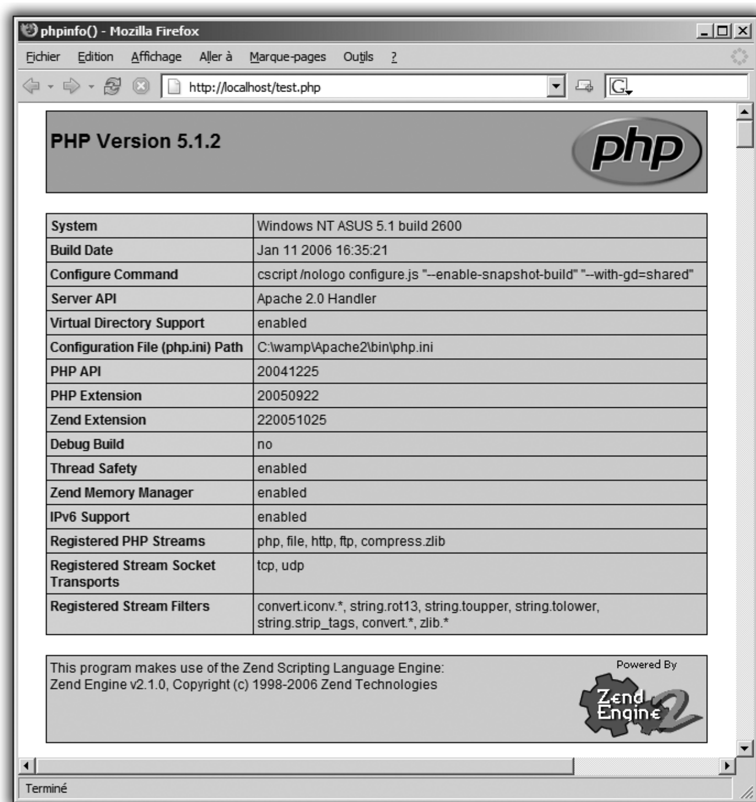


Figure 3.4 : *Informations retournées par la fonction phpinfo()*

Comme vous le voyez, les fonctions ont des rôles très variés. Une simple fonction peut réaliser des tâches plus ou moins complexes.



Le fichier *test.php*

Plutôt que de créer pour chacun des petits exemples un nouveau fichier (*test1.php*, *test2.php*, etc.), le fichier *test.php* sera réutilisé à chaque fois.

Au sein d'un même script, les blocs de code peuvent être multiples et du « code » HTML peut s'y intercaler. Toute zone non comprise entre les balises `<?php` et `?>` est considérée comme du HTML (ou tout du moins comme du texte brut). Elle est par conséquent directement retournée :

Listing 3-5 : PHP + HTML

```
<?php
print("premier bloc PHP");
?>

<hr>
partie <b>HTML</b>
<hr>

<?php
print("deuxième bloc <i>PHP</i>");
?>
```

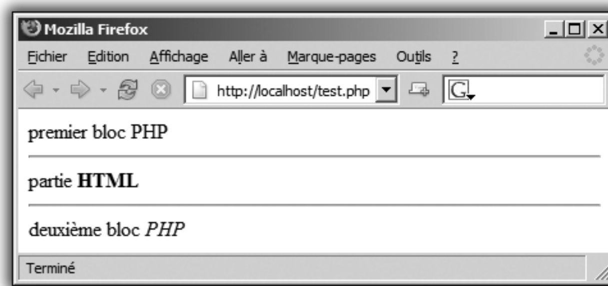


Figure 3.5 : *Du PHP et du HTML au sein du même script*

Au sein de ce script, les deux blocs suivants correspondent à du code PHP :

```
<?php
print("premier bloc PHP");
?>
<?php
print("deuxième bloc <i>PHP</i>");
?>
```

Ces blocs seront remplacés par le résultat de leur interprétation.

La partie suivante correspond quant à elle à du simple code HTML qui est « retourné » tel quel par le serveur.

```
<hr>
partie <b>HTML</b>
<hr>
```

Vous pouvez remarquer que dans le deuxième bloc de code sont affichées des données qui contiennent des balises HTML permettant la mise en italique d'un texte (`<i>PHP</i>`). C'est non seulement tout à fait autorisé, mais de plus très courant en PHP.



HTML et PHP

Il serait tout à fait envisageable qu'un fichier *.php* ne contienne que du HTML. La page serait retournée sans aucune modification. L'intérêt est cependant bien mince car vous ajoutez, dans ce cas, une étape entre le serveur web et vous. En effet, bien que la page ne contienne aucun bloc d'instructions, l'interpréteur sera quand même mis à contribution.

En affichant les sources de la page, vous obtenez la preuve que le navigateur a bien reçu le résultat de l'interprétation du code par PHP et que la partie HTML n'a pas été modifiée.

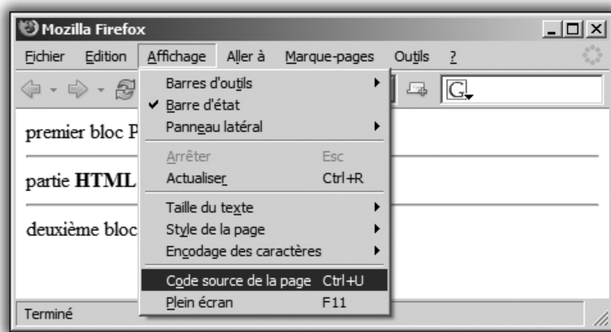


Figure 3.6 : *Sources*

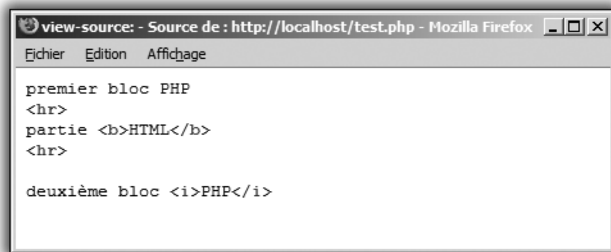


Figure 3.7 : *Les instructions PHP ont bien été interprétées*

3.2. Les commentaires

Un bloc de code peut contenir des commentaires. Ces derniers peuvent être ajoutés au code de différentes manières :

Listing 3-6 : Différentes syntaxes permettant d'insérer des commentaires

```
<?php

/* commentaires hérités
du C */

// commentaires hérités
// du C++

# commentaires hérités
# du SHELL

print("code commenté"); // commentaire de fin de ligne

?>
```

Ajouter des commentaires ne ralentit en rien l'exécution de votre code. Au moment de l'interprétation, PHP va tout simplement supprimer ces zones commentées.

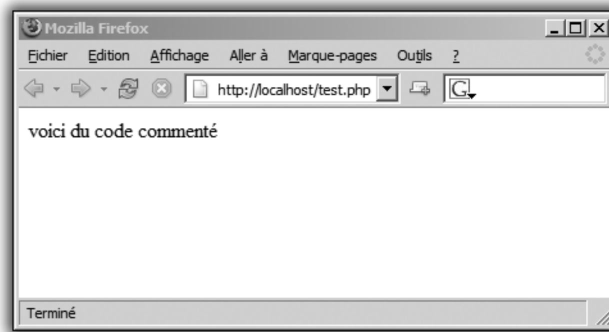


Figure 3.8 : *Les parties commentées n'apparaissent pas*

Poussons le vice un peu plus loin en générant un commentaire HTML à l'aide d'un script PHP. Les commentaires HTML doivent être placés entre <!— et —> :

Listing 3-7 : Commentaire HTML

```
<?php

print("voici un commentaire HTML : ");
print("<!-- message commenté -->");

?>
```

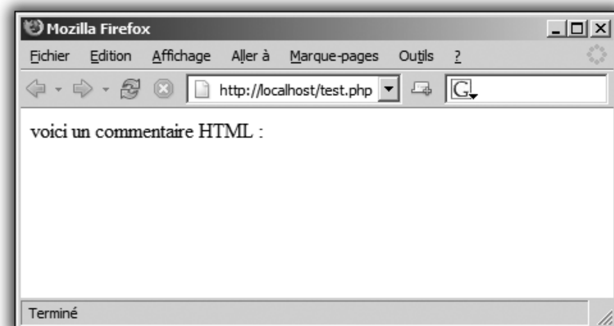


Figure 3.9 : *Code avec commentaires*

Bien qu'invisible dans le navigateur, le commentaire HTML est cependant bien présent dans les sources de la page.

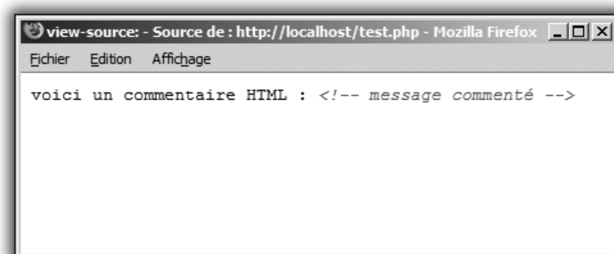


Figure 3.10 : *Sources*

3.3. Les variables

Quel que soit le langage de programmation, l'élément principal d'un programme est la variable. Une variable est un élément qui peut prendre différentes valeurs au cours de l'exécution d'un programme.

En PHP, les noms des variables sont précédés du caractère `$` : `$abc` correspond à la variable `abc`.

`$abc` est ici le nom de la variable, à ne pas confondre avec ce que contient la variable (`$abc` peut par exemple contenir la valeur `3`).

Précisément, pour donner la valeur `3` à la variable `$abc`, écrivez l'affectation suivante :

Listing 3-8 : Affectation de la valeur 3 à la variable $abc

```
$abc = 3;
```



Notation des exemples

Pour de tout petits exemples, comme celui-ci, ne correspondant en fait qu'à une sous-partie de script, les balises `<?php` et `?>` ne sont pas notées. Si vous voulez tester ce morceau de code, il convient bien évidemment de les ajouter.

Le contenu d'une variable peut être obtenu en y faisant simplement référence :

Listing 3-9 : Affectation de différentes valeurs

```
valeur de la variable abc =
<?php
print($abc);
?>

<br/>valeur de la variable abc =
<?php
$abc = 3;
print($abc)
?>

<br/>valeur de la variable abc =
<?php
$abc = 12;
print($abc)
?>
```

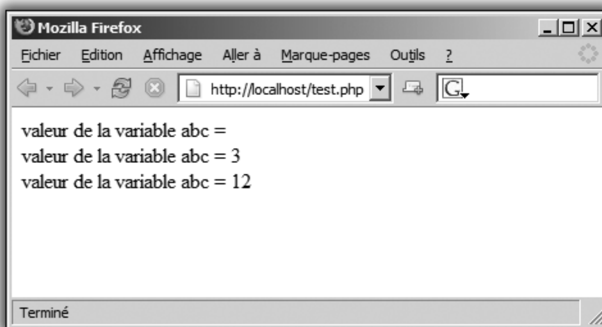


Figure 3.11 : *Différentes affectations de variables*

Vous constatez qu'avant qu'une valeur ne soit affectée à la variable, celle-ci n'existe pas. Elle peut ensuite prendre différentes valeurs lors de l'exécution du script.

Il faut aussi noter que la valeur d'une variable est conservée d'un bloc de code à l'autre.

La valeur d'une variable peut être affectée tout naturellement à une autre variable avec l'opérateur d'affectation = :

Listing 3-10 : La variable $abc contient la valeur de $d (2) à l'issue de l'affectation

```
<?php
$abc = 1;
$d = 2;
$abc = $d;
?>
```

Il existe différents moyens permettant d'affecter une valeur à une variable :

- soit directement : `$abc = 2;`;
- soit en lui faisant recevoir le résultat d'une opération : `$abc = 1 + 3;`;
- soit en lui faisant recevoir le résultat d'une fonction : `$abc = pow(2,4);` (c'est-à-dire la puissance 4 de 2).

Listing 3-11 : Différents types d'affectations

```
1- abc =
<?php
$abc = 2;
print($abc);
?>

<br/>2- abc =
<?php
$abc = 3 * 2;
print($abc)
?>

<br/>3- abc =
<?php
$abc = $abc + 1;
print($abc)
?>

<br/>4- abc =
<?php
```

```
$abc = pow(2,4);
print($abc)
?>

<br/>5- abc =
<?php
$abc = pow(2,4) - 7;
print($abc)
?>
```

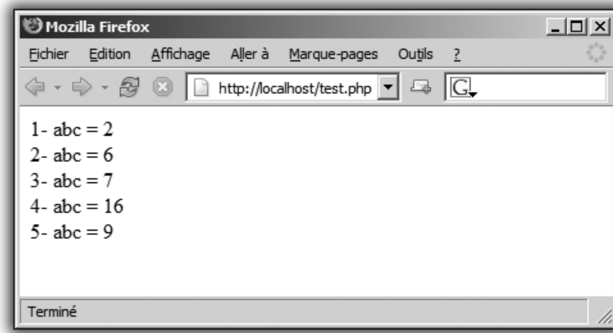


Figure 3.12 : *Différents types d'affectations*

L'instruction `$abc = $abc + 1` indique que la variable `$abc` est égale à la valeur de la variable `$abc` plus 1. Il est possible d'écrire plus succinctement cette affectation : `$abc++`.

L'opérateur ++ derrière une variable incrémente cette variable et inversement, l'opérateur –– décrémente la valeur.

```
abc =
<?php
$abc = 4;
$abc--;
print($abc);
?>
```



Certains noms de variables sont interdits, reportez-vous aux** Annexes **pour obtenir plus de précisions sur ce sujet.

Il existe une notation alternative et plus complexe qui permet d'avoir accès aux variables : `${'nom_de_la_variable'}`.

```
$x = 0;
${'x'} = 10; // assigne la valeur 10 à la variable x
print(${'x'}); // affiche : 10
```

```
print($x); // affiche : 10
$x = 5; // assigne 5 à la variable x
print(${"x"}); // affiche : 5

$y = "x";
print(${$y}); // affiche 5 car $x vaut 5
```

Ces deux notations sont donc complètement équivalentes ; la deuxième peut cependant se révéler très utile lorsque vous devez créer ou récupérer des variables à l'orthographe non standard ou des variables dont le nom est lui-même « variable ». L'instruction suivante est tout à fait valable, malgré la présence d'un espace dans le nom de la variable :

```
${"ma variable"} = "bonjour monde";
```



Majuscules et minuscules

Le nom des variables est sensible à la casse (*case sensitive*). Cela signifie que PHP fait une différence entre majuscules et minuscules. Une variable `$a` sera donc différente de la variable `$A` et pourra donc cohabiter avec la première en disposant de sa propre valeur.

3.4. Les constantes

Les constantes peuvent être assimilées à des variables dont le contenu ne peut être pas modifié durant l'exécution du programme. La déclaration d'une constante fait appel à la fonction `define()` dont le premier argument correspond au nom de la constante, et le second, à sa valeur.

Listing 3-12 : Création de la constante VERSION_SITE

```
define("VERSION_SITE","v3.1");
```

À la différence des variables, les constantes n'ont pas à être précédées du caractère `$`.

Listing 3-13 : Utilisation de constantes au sein d'un site

```
<?php

define("NOM_SITE","Cool Site");
print(NOM_SITE);

define("VERSION_SITE",3.1);
$a = VERSION_SITE + 1;

?>
```

PHP dispose d'un certain nombre de constantes définies par défaut. La constante `PHP_VERSION` contient par exemple la version de l'interpréteur PHP.

Listing 3-14 : Utilisation d'une constante par défaut

```
<?php

print("Version de PHP : ");
print(PHP_VERSION);

?>
```

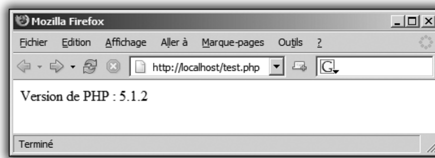


Figure 3.13 : *Version de PHP*

L'exécution du script suivant vous permet d'obtenir la liste de toutes les constantes définies par défaut par PHP.

Listing 3-15 : Plus de 700 constantes définies par PHP

```
<pre>
<?php
print_r(get_defined_constants());
?>
</pre>
```

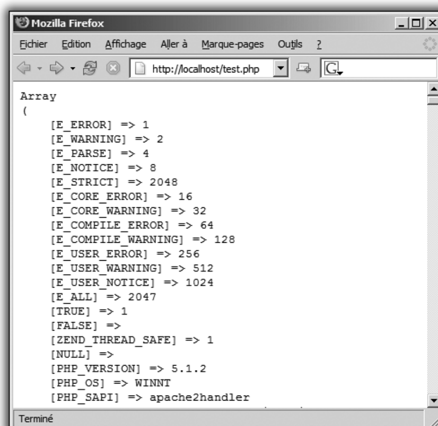


Figure 3.14 : *Plus de 700 constantes*



Norme d'écriture

Les constantes sont généralement écrites en majuscule afin de les différencier des variables. Ce standard est partagé par une grande majorité des langages de programmation.

3.5. Les types de données

En informatique, les données sont souvent typées, en ce sens qu'elles contiennent une certaine catégorie d'information. Il existe globalement deux principaux types de données : les variables contenant des chiffres et les variables contenant des lettres.



Les tableaux, qui correspondent également à un type de donnée, sont présentés dans un chapitre suivant.

PHP rend les choses extrêmement simples au niveau des types car il n'impose pas d'associer explicitement un type à une variable (dans la plupart des autres langages, les variables doivent être associées à un type dès leur initialisation au début du programme). Si vous affectez `3` à `$abc`, celle-ci devient *de facto* de type numérique. Si en revanche vous lui affectez la phrase `"bonjour monde"`, elle devient ce que l'on appelle une « chaîne de caractères » (souvent appelée *string*).

Arrêtons-nous quelque temps sur ces deux principaux types de données.

Les données numériques

Vous trouvez, parmi les données numériques, les nombres entiers (`2`, `4`, `233`, `–12`) et les nombres flottants (`0.5`, `23.8`, `–123.4`).



Écriture des nombres flottants

Notez que le caractère séparant la partie entière de la partie décimale est le point. En utilisant une virgule, vous générez une erreur. Il s'agit de la norme anglo-saxonne qui est utilisée par tous les langages de programmation.

Il existe un certain nombre d'opérateurs mathématiques qui peuvent être utilisés avec les numériques :

Tableau 3.1 : Les différents opérateurs mathématiques	
Opérateur	**Rôle**
+	L'addition
–	La soustraction
*	La multiplication
/	La division
%	Le modulo

Voyez l'utilisation de tous ces opérateurs dans un même exemple :

Listing 3-16 : Différents opérateurs mathématiques

```
<?php
$a = 10 ;
$b = 5 ;
$c = 2;

$x = $a + $b; // la variable $x contient 15
$x = $x - $c ; // $x vaut 13
$x = $x * $x ; // $x vaut 169
$x = ($a / $b) + $c; // $x vaut 4
$x = 11 % $a ;
// $x vaut 1,
// le modulo correspond au reste de la division 11 / 10
?>
```

L'ordre de priorité des opérateurs doit être respecté : les opérations `* / %` sont traitées avant les opérations `+ –`.

L'expression `10 – 2 * 4` vaut ainsi `2` et non `32`. La multiplication est en effet réalisée avant la soustraction.

Quand PHP doit interpréter la ligne `$abc = 16 – 12 / 6 * 5 + 1;`, il évalue l'expression de la manière suivante :

```
16 - 2 * 5 + 1
16 - 10 + 1
6 + 1
7
```

Il peut donc être préférable d'utiliser des parenthèses pour limiter les risques :

```
$abc = 16 - ( ( 12 / 6 ) * 5 ) + 1 ;
```

Afin de gagner du temps, des formes compactes existent pour certaines opérations. Ainsi, `$abc = $abc + 2 ;` est équivalent à `$abc += 2 ;`

De la même manière, les raccourcis suivants sont tout à fait valides : –=, `*=`, `/=` et `%=`.

Il existe un type de numérique à part : les booléens. Seules deux valeurs booléennes existent : `true` (vrai), `false` (faux).

```
<?php
$var = true ; // $var contient la valeur booléenne true
?>
```

Il est courant que la valeur `0` soit équivalente à `false` et que la valeur `1` soit équivalente à `true`. Cet abus peut cependant se révéler dangereux lors de certains tests. Nous y ferons d'ailleurs mention plus loin.

Les chaînes de caractères

Une chaîne de caractères (souvent appelée *string*) doit être délimitée par des guillemets (") ou des primes (').

Les guillemets et les primes

Lorsque les caractères de délimitation sont des guillemets ("), la chaîne de caractères peut contenir des caractères ainsi que des variables. La variable est alors remplacée par son contenu.

Listing 3-17 : Une variable dans une chaîne de caractères

```
<?php
$n = 2;
$s = "valeur de la variable n = $n";
print($s); // affiche : "valeur de la variable n = 2"
?>
```

Quand les primes (') sont utilisées comme caractères de délimitation, les variables ne sont plus remplacées par leur contenu :

```
<?php
$n = 2;
$s = 'valeur de la variable n = $n';
```

```
print($s); // affiche : 'valeur de la variable n = $n'
?>
```



Bien utiliser les signes

En utilisant la prime ('), PHP sait qu'il n'a pas à remplacer la présence d'une éventuelle variable par son contenu. Il affiche directement et sans traitement la chaîne de caractères. Il est donc préférable, pour optimiser votre code, d'utiliser les guillemets quand une chaîne est vierge de toute variable. Par exemple, `'bonjour monde'` est préférable à `"bonjour monde"`.

L'affichage d'une chaîne de caractères peut également être réalisé avec la fonction `echo()`. Les syntaxes suivantes pourront être trouvées indifféremment dans la littérature :

- `echo "bonjour";`
- `echo("bonjour");`
- `print("bonjour");`
- `print "bonjour";`



Syntaxe spéciale de `echo`

`echo` permet l'utilisation d'une syntaxe spéciale : `echo $var1, $var2;` pour afficher la `$var1` puis `$var2`. Aucune limitation n'existe au niveau du nombre de variables à afficher.

Échappement de caractères

Une question peut alors se poser : comment une chaîne de caractères peut contenir le guillemet (") quand ses délimiteurs sont précisément des guillemets ?

Il est dans ce cas nécessaire d'utiliser un caractère dit d'échappement : `\`. Ce caractère est nommé « barre oblique inversée » ou *antislash* (également *backslash*). Pour obtenir la phrase `bonjour " monde`, il est nécessaire d'écrire `"bonjour \" monde"`. Le principe est le même pour la prime (') avec des strings délimitées par des primes :

Listing 3-18 : Échappement de caractères

```
print("bonjour \" monde"); // affiche : bonjour " monde
print('bonjour \' monde'); // affiche : bonjour ' monde
print('bonjour \" monde'); // affiche : bonjour \" monde
```

Les caractères (\) et ($) ont aussi besoin d'être précédés d'une barre oblique inversée pour être affichés dans une chaîne entre guillemets :

```
$i = 2 ;
print("\\ \$i vaut $i"); // affiche : "\ $i vaut 2"
```

L'opérateur de concaténation

Il existe un opérateur pour les chaînes de caractères : l'opérateur de concaténation. Il permet de réunir deux chaînes : `$str = $str1 . $str2 ;` et signifie que la variable `$str` contient la variable `$str1` suivie de `$str2`.

Listing 3-19 : L'opérateur de concaténation

```
<?php
$var1 = "ui";
$var2 = "le temps est " .  "beau aujourd'h" . $var1;
print($var2); // affiche : "le temps est beau aujourd'hui"
print("cou" . 'cou'); // affiche : "coucou"
?>
```

La version raccourcie existe aussi : `.=`.

`$abc = $abc . " coucou" ;` est l'équivalent de `$abc .= " coucou";`.

L'opérateur de concaténation est également très utile pour améliorer la lisibilité d'une chaîne de caractères très longue.

Listing 3-20 : $var1 et $var2 sont rigoureusement identiques

```
<?php
$var1 = "Les 7 péchés capitaux sont : la paresse,
✂ l'orgueil, la gourmandise, la luxure, l'avarice, la
✂ colère, l'envie.";
$var2 = "Les 7 péchés capitaux sont : ".
        "la paresse, l'orgueil, la gourmandise, la luxure, ".
         "l'avarice, la colère, l'envie.";
?>
```

En informatique, les chaînes de caractères sont souvent appelées *string*. Vous trouverez donc souvent dans les exemples suivants la variable `$str` pour définir une variable de type chaîne de caractères.



Appellation des variables

Il est important, en programmation, de donner des noms pertinents aux variables. Quand vous voulez qu'une variable contienne le nom d'une ville, il est préférable de nommer la variable `$ville` plutôt que `$x`. N'hésitez donc pas à donner des noms « longs » : `$portefeuille_client` est préférable à `$prtfcl`, même s'il peut paraître rébarbatif d'écrire une variable aussi longue. Comme nous l'avons précédemment dit, votre code devra peut-être être repris après plusieurs mois, et pas obligatoirement par vous !

Le type NULL

Ce type, composé d'un seul élément (`NULL`), permet d'indiquer qu'une variable n'a pas de valeur.

Listing 3-21 : Une fois utilisée, la variable $str est passée à NULL

```
<?php
$str = "coucou";
print($str);
$str = NULL;
?>
```

Changement de type

PHP a ceci de sympathique qu'il permet aux variables de changer de type si vous leur affectez des données de types différents au cours du programme :

Listing 3-22 : Changement de type

```
<?php
$abc = "1"; // $abc est une string contenant la chaîne "1"
$abc += 2; // $abc est maintenant un numérique contenant
✂ la valeur 3
?>
```

Dans cet exemple, la variable est d'abord une chaîne de caractères contenant la chaîne `"1"`. Après l'opération d'incrémentation, elle devient de type numérique et contient la valeur `3`.

PHP propose également une opération de transtypage pour convertir les types de données. Cette opération, qui porte également le nom de « *cast* », utilise la syntaxe suivante :

```
$s = (string) 1; // $s contient maintenant la chaîne "1"
$n = (int) $s; // $n contient 1
$b = (bool) $n; // $b contient TRUE
$n = (int) 1.8; // $n contient 1
$b = (bool) -3; // $b contient TRUE
$b = (bool) 0; // $b contient FALSE
$f = (float) " 1.45 "; // $f contient 1.45
```



Autres types

Le transtypage fonctionne également avec les autres types de données étudiés dans la suite de l'ouvrage : les tableaux (`array`) et les objets (`object`).

Au final, PHP permet de ne pas avoir à se tracasser avec les problèmes de typage.

3.6. Les structures de contrôle

Tous ces exemples s'exécutaient jusqu'à maintenant linéairement, ligne après ligne. Cependant, comme dans la vie courante, il est rare que les choses se passent aussi simplement : nous réalisons certaines choses uniquement dans certains cas et, dans d'autres circonstances, nous répétons les mêmes choses plusieurs fois. L'équivalent informatique de ces événements est la structure de contrôle.

Vous disposez, avec PHP, de toutes les structures de contrôle standard. Celles-ci sont généralement regroupées en deux catégories…

Les conditions :

```
SI test est vrai ALORS FAIRE action1 SINON FAIRE action2

FAIRE DANS LE CAS 1 action1, DANS LE CAS 2 action 2 etc.
```

Les boucles :

```
FAIRE action TANT QUE test est vrai

TANT QUE test est vrai ALORS FAIRE action

POUR TOUS LES CAS SUIVANTS FAIRE action
```

Les conditions

Nous présenterons dans cette partie les expressions `if… else`, `if… elseif… else` et `switch… case`.

IF ELSE

L'équivalent de `SI test est vrai ALORS FAIRE action1 SINON FAIRE action2` est l'instruction `if… else` (*if* signifie « si » et *else* signifie « ou bien »).

Écrivez un petit programme qui affiche `"i est plus grand que 5"` si la variable `$i` est plus grande que 5 et la phrase `"i est plus petit que 5"` dans le cas contraire.

Listing 3-23 : Le premier test

```
<?php

$i = 4;

if ($i > 5)
  print("i est plus grand que 5");
else
  print("i est plus petit que 5");

?>
```

Dans ce cas, `"i est plus petit que 5"` apparaît à l'écran car vous avez initialisé la variable `$i` à `4`.

Les tests sont réalisés avec des opérateurs de comparaison.

Tableau 3.2 : *Les opérateurs de comparaison*

Opérateur	Résultat
`$a == $b`	Vrai si `$a` est égal à `$b`
`$a === $b`	Vrai si `$a` est égal à `$b` et si ces deux variables sont de même type
`$a != $b`	Vrai si `$a` est différent de `$b`
`$a !== $b`	Vrai si `$a` est différent de `$b` (en type ou en valeur)
`$a > $b`	Vrai si `$a` est supérieur à `$b`

Tableau 3.2 : *Les opérateurs de comparaison*	
Opérateur	**Résultat**
`$a < $b`	Vrai si `$a` est inférieur à `$b`
`$a >= $b`	Vrai si `$a` est supérieur ou égal à `$b`
`$a <= $b`	Vrai si `$a` est inférieur ou égal à `$b`

Il est important de comprendre que ces opérateurs renvoient une valeur booléenne, `false` ou `true`, suivant le résultat de la comparaison. Le test est en fait réalisé sur cette valeur de retour.

Écrire `if ($a > $b)` est en fait la même chose qu'écrire `if (($a > $b) == true)`.



`if ($i)`

L'expression `if ($i)` est identique à `if ($i != NULL)`. Cette version condensée est très souvent utilisée en informatique.

Dans cet exemple, une seule action est réalisée, dans un cas (`if`) comme dans l'autre (`else`). Si plusieurs actions doivent être réalisées, celles-ci sont groupées entre des balises `{ }` pour former un groupe d'instructions.

Modifiez votre script afin qu'il affiche deux lignes :

```
<?php

$i = 4;

if ($i > 5)
{
  print("nous sommes dans le IF ... ");
  print("i est plus grand que 5");
}
else
{
print("nous sommes dans le ELSE ... ");
  print("i est plus petit que 5");
}
?>
```



Les balises `{ }`

Alors qu'elles sont facultatives quand il n'y a qu'une instruction, elles deviennent obligatoires dès qu'il y en a au moins deux.

Les tests peuvent se faire aussi bien sur des numériques :

```
if ($prix == 3.5)
```

… que sur des chaînes de caractères :

```
if ($prenom == "paul")
```

Attention, cependant, à ne pas se méprendre sur les opérateurs de supériorité et d'infériorité avec des chaînes de caractères !

Ainsi, le test `if ($code > "9AEXB3")` n'est pas un test sur les longueurs respectives des deux chaînes. Il s'agit plutôt de comparer une à une les valeurs ASCII des caractères composant les deux chaînes. Chaque caractère possède en informatique un code ASCII. Celui du caractère `'a'` est par exemple 97 et celui de `'9'` est 57 (la fonction `ord()` peut être utilisée pour trouver la valeur ASCII d'un caractère : `ord("A")` retourne 98). De ce fait, `"abc"` est supérieur à `"9AEXB3"` (car `ord('a')` est supérieur à `ord('9')`) et `"def"` est supérieur à `"dc"` (car `ord('e')` est supérieur à `ord('c')`).

Pour l'instant, les tests étudiés sont simples car vous ne testez qu'une seule valeur. Si vous devez écrire le test suivant :

```
SI la personne a plus de 18 ans ALORS écrire "homme
majeur" SI elle est de sexe masculin
```

… vous écrirez, dans l'état de vos connaissances actuelles :

```
<?php

$age = 22;
$sexe = "masculin";

if ($age > 18) {
  if ($sexe == "masculin")
    print("homme majeur");
}

?>
```

La façon logique d'exprimer ce test est :

```
SI la personne a plus de 18 ans ET qu'elle est de sexe
masculin ALORS écrire "homme majeur"
```

Il s'agit là d'une double condition. L'opérateur (`&&`) peut être utilisé pour signifier l'opérateur ET :

```
<?php

$age = 22;
$sexe = "masculin";

if (($age > 18) && ($sexe == "masculin"))
  print("homme majeur");

// affiche bien "homme majeur"
// car : (22 > 18) ET (masculin == masculin)

?>
```

L'opérateur (`&&`) est appelé un « opérateur logique ». Il en existe d'autres qui vont vous permettre de réaliser des tests plus ardus.

Tableau 3.3 : *Opérateurs logiques*

Opérateur	Résultat
`$a and $b`	Vrai si `$a` ET `$b` sont vraies
`$a && $b`	Vrai si `$a` ET `$b` sont vraies (identique à and)
`$a or $b`	Vrai si `$a` OU `$b` sont vraies
`$a || $b`	Vrai si `$a` OU `$b` sont vraies (identique à or)
`$a xor $b`	Vrai si `$a` OU `$b` sont vraies, mais pas les deux
`! $a`	Vrai si `$a` est fausse

À partir de ces opérateurs logiques, il est possible d'écrire le test suivant :

```
SI la variable $a OU la variable $b est supérieure à 4,
ET que la variable $c n'est pas inférieure ou égale à 2
ALORS écrire OK
```

… qui devient :

```
if ((($a > 4) or ($b > 4)) and !($c <= 2)) print("OK");
```

Il est souvent possible de simplifier une expression : `!($c <= 2)` équivaut à `($c > 2)`. En effet, dire que `$c` n'est pas inférieur ou égal à `2` est une façon complexe de dire que `$c` est supérieur à `2`.

Il est néanmoins préférable généralement, au début, de transcrire terme à terme une condition plutôt que d'essayer de la simplifier à outrance.

Un bon réflexe consiste aussi à insister sur les parenthèses afin de regrouper ensemble les conditions interdépendantes. Les priorités, comme pour les opérateurs mathématiques, peuvent faire des ravages. Le test suivant :

```
($b > 3) || ($c > 3) && ($c < 10)
```

… ne signifie pas :

```
$b OU $c supérieures à 3 ET $c inférieure à 10
```

… mais :

```
$b supérieure à 3 OU $c comprise entre 3 ET 10
```

Dans cette mesure, il est préférable d'écrire en utilisant une paire de parenthèses supplémentaires pour ne rien risquer :

```
($b > 3) || (($c > 3) && ($c < 10))
```



Les annexes contiennent un tableau présentant l'ordre de priorité des différents opérateurs.

IF… ELSEIF…

Essayez maintenant d'écrire un autre test :

```
si la couleur est rouge, jaune ou bleue, écrire
"primaire"
si la couleur est noire, écrire "noire"
si la couleur est blanche, écrire "blanche"
sinon écrire "mélange"
```

Votre première proposition pourrait être celle-ci :

```
if (($couleur=="rouge") || ($couleur=="jaune") ||
    ($couleur=="bleue"))
  print("primaire");
```

```
if ($couleur == "noire") print("noire");

if ($couleur == "blanche") print("blanche");
else print("mélange");

}
```

Celle-ci serait évidemment complètement erronée car, si vous supposez que la « couleur » est `rouge`, deux messages seraient alors affichés : `"primaire"` et `"mélange"`.

Vous êtes donc obligé d'emboîter les `if… else` :

```
if (($couleur=="rouge") || ($couleur=="jaune") ||
   ($couleur=="bleue"))
  print("primaire");
else
{
  if ($couleur == "noire") print("noire");
    else
  if ($couleur == "blanche") print("blanche");
    else print("mélange");
}
```

Il s'agit en fait d'un SI… SINON SI… SINON SI… SINON. PHP propose, pour ce genre de cas, la méthode suivante : `if… elseif… elseif… else`.

```
<?php

if (($couleur=="rouge") || ($couleur=="jaune") ||
    ($couleur=="bleue"))
  print("primaire");
elseif ($couleur == "noire")
  print("noire");
elseif ($couleur == "blanche")
  print("blanche");
else
  print("mélange");

?>
```

SWITCH

Supposez désormais que vous deviez écrire le test suivant :

```
si la couleur est rouge, écrire "R"
```

```
si la couleur est bleue, écrire "B"

si la couleur est jaune, écrire "J"

sinon écrire "?"
```

Dans ce cas, il est dommage d'utiliser un `if… elseif…`. Imaginez, en effet, que la couleur soit jaune : il faudrait alors tester si la couleur est rouge, puis tester si la couleur est bleue et enfin tester si la couleur est jaune. Dans ce type de cas, où l'action à réaliser ne dépend que de la valeur d'une variable, il est préférable d'utiliser le `switch` :

```
switch ($couleur)
{
  case "rouge":
    print("R");
    break;
  case "bleue":
    print("B");
    break;
  case "jaune":
    print("J");
    break;
  default:
    print("?");
    break;
}
```

Il peut y avoir autant de `case` que nécessaire, mais il ne peut y avoir qu'un seul `default` (il n'est cependant pas obligatoire).

Un `case` peut contenir plusieurs instructions ; il faut alors les placer entre un `case` et un `break` :

```
switch ($couleur)
{
  case "rouge":
    print("couleur : ");
    print("R");
    break;
  case "bleue":
etc.
```

Les boucles

Un des principaux avantages de l'informatique, en général, est de permettre d'automatiser des tâches fastidieuses. Imaginez que vous vouliez écrire tous les entiers inférieurs à 5.

Il serait dommage de devoir écrire :

```
print("0<br/>");
print("1<br/>");
print("2<br/>");
print("3<br/>");
print("4<br/>");
```

L'informatique a créé, pour ce genre de besoin, la notion de boucle.

WHILE

`WHILE` est la boucle qui permet d'écrire ceci :

```
TANT QUE test vrai FAIRE action
```

À partir de là, il devient évident pour cet exemple que le test va être réalisé sur une variable (`$i < 5`) et que l'action va consister à l'afficher et à l'incrémenter :

```
$i = 0;
while ($i < 5)
{
  print("$i<br/>");
  $i++;
}
print("<br/>fin de la boucle");
```

Que se passe t-il au niveau de l'interpréteur PHP :

- la variable `$i` prend la valeur `0`
- nous entrons dans la boucle
- est-ce que `$i < 5` ? oui : le bloc d'expressions du `while` est alors exécuté
- affichage de `$i` (`0`) et augmentation de `1` de la valeur de `$i`
- nous remontons alors au niveau du test du `while`
- `$i` est-elle inférieure à `5` ? oui (elle vaut `1`), alors on exécute le bloc
- affichage de `$i` (`1`) et incrémentation de `$i`…
- nous continuons ainsi jusqu'au moment où nous passons la valeur de `$i` à `5`
- nous remontons alors au niveau du test, `$i` n'est plus strictement inférieure à `5`

– nous sortons alors de la boucle et passons à l'instruction suivant le bloc du `while`, c'est-à-dire l'affichage de "*fin de la boucle*"



Le danger du `while`

Faites attention, quand vous travaillez avec une boucle `while`, à ne pas vous retrouver dans une boucle infinie. Cela aurait été le cas, si vous aviez oublié la ligne `$i++`. En effet, la variable aurait gardé la valeur `0` et la boucle `while` aurait affiché `"0"` indéfiniment.

Comme pour les conditions, le test peut être complexe :

```
$i = 1;
while (($n != 121) && ($i < 100))
{
  $n = $i * $i;
  print("i = $i , n= $n<br/>");
  $i++;
}
```

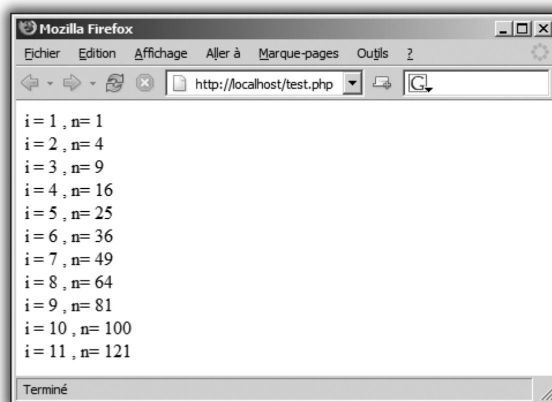


Figure 3.15 : *Boucle while*

Dans cet exemple, la condition d'arrêt de la boucle est double : la variable `$n` doit être différente de `121` et la variable `$i` doit être inférieure à `100`.

DO… WHILE

Une variante existe au `while` : le `do… while`. Il ne s'agit plus d'un `tant que… faire`, mais d'un `faire… tant que`. Dans ce cas, le code de la boucle est donc au moins exécuté une fois :

```
$i = 6;
do
{
  print($i);
}
while ($i < 5);
```

Bien que `$i` contienne la valeur `6` et que le test soit `$i < 5`, le code est bien exécuté une fois.

FOR

La syntaxe de la boucle `for` est la suivante :

```
for (expression 1; expression 2; expression 3)
```

- `expression 1` est l'instruction initiale exécutée avant la première itération ;
- `expression 2` est le test réalisé au début pour chaque itération ;
- `expression 3` est l'instruction exécutée à la fin de chaque itération.

Ainsi, l'équivalent de cette boucle `while` :

```
$i = 0;
while ($i <= 10)
{
  print("$i <br/>");
  $i++;
}
```

… est la boucle `for` suivante :

```
for ($i = 0; $i <= 10; $i++)
{
  print("$i <br/>");
}
```

Vous souhaitez maintenant écrire les nombres pairs inférieurs à 50 :

```
for ($i = 0; $i <= 50; $i = $i + 2)
{
  print("$i <br/>");
}
```

La boucle `for` permet donc de condenser les principaux éléments d'une boucle sur une seule et même ligne.

Toutes les parties de la boucle `for` peuvent être vides. Ainsi, les deux boucles suivantes sont équivalentes :

Listing 3-24 : Version 1

```
for ($i = 0; $i <= 10; $i++)
{
  print("$i <br/>");
}
```

Listing 3-25 : Version 2

```
$i = 0;
for (;$i <= 10;)
{
  print("$i <br/>");
  $i++;
}
```

L'intérêt de cette notation alambiquée est cependant assez mince.

BREAK et CONTINUE

Il est parfois nécessaire de terminer une boucle en plein milieu de son exécution ; l'instruction `break` est alors utilisée.

La boucle suivante permet d'arrêter l'exécution du `for` dès que la variable d'incrémentation passe à la valeur 7.

Listing 3-26 : Le break nous permet de sortir de la boucle

```
for ($i=0;$i<=10;$i++) {
  if ($i==7) {
    print("<i>i contient 7, nous sortons de la
    ✂ boucle.</i>");
    print("<br/><br/>");
    break;
  }
  print("$i<br/>");
}
print("<b>fin de la boucle.</b>");
```

(voir Figure 3.16)

Le cousin du `break` est le `continue`. Il permet de forcer le passage à l'itération suivante à n'importe quel niveau d'une boucle.

L'instruction `continue` permet dans l'exemple suivant de sauter l'affichage de la valeur 7 :



Figure 3.16 : *Sortie en plein milieu de boucle*

Listing 3-27 : Nous sautons le traitement de la valeur 7

```
for ($i=0;$i<=10;$i++) {
  if ($i==7) {
    print("<i>i contient 7, passage à l'itération
    ✂ suivante.</i>");
    print("<br/>");
    continue;
  }
  print("$i<br/>");
}
print("<br/><b>fin de la boucle.</b>");
```



Figure 3.17 : *Instruction continue*

Ces deux instructions peuvent aussi bien être utilisées dans le cadre d'un `for`, d'un `while` ou d'un `switch`.

3.7. Organisation du code

PHP propose différentes méthodes pour organiser vos développements : la définition de fonctions utilisateur et l'inclusion de fichiers. Ces méthodes visent avant à tout à :

- améliorer la lisibilité de vos sources ;
- éviter la duplication de code.



Les objets, qui permettent également de mieux architecturer votre code, sont présentés dans un chapitre à part.

Les fonctions

Vous savez désormais que PHP permet de faire appel à des fonctions. Les exemples précédents vous ont ainsi permis d'illustrer l'utilisation de la fonction `print()` dont le rôle est d'afficher les données qui lui sont transmises.

La plupart des fonctions reçoivent des paramètres et retournent une valeur qui en dépend : on parle d'« arguments d'une fonction ». Par exemple, la fonction `print()` prend comme paramètre une chaîne de caractères ou un chiffre. Sa signature est donc celle-ci : `print($str)`.

La fonction de puissance prend quant à elle, deux arguments, à savoir le nombre et la puissance à laquelle il doit être élevé : `pow($n,$p)`.

```
$resultat = pow(2,4);
// $resultat contient 16
// qui correspond bien à 2 exposant 4
```

Ces deux fonctions, comme plusieurs centaines d'autres, sont incluses par défaut dans PHP. Vous pouvez y faire appel à n'importe quel moment dans votre code.

En plus de cette « bibliothèque » de fonctions, PHP vous permet également de définir vos propres fonctions.

Définir une fonction

Supposez que vous vouliez qu'une fonction renvoie le double d'une valeur. Il faut tout d'abord lui donner un nom, `double()`, et décider de combien d'arguments elle a besoin ; a priori, un seul. La syntaxe pour déclarer une fonction en PHP est la suivante :

```
function double($n)
{
  $resultat = $n * 2;
  return $resultat;
}
```

Nommer l'argument `$n` n'est pas du tout obligatoire. Vous auriez tout à fait pu écrire la fonction ainsi :

```
function double($nimportequoi)
{
  $resultat = $nimportequoi * 2;
  return $resultat;
}
```

Il est cependant souvent intéressant de donner un nom d'argument en rapport avec le type de l'argument. Ainsi, si la fonction nécessite comme paramètre un booléen, il peut être judicieux de choisir `$bool` ; et s'il s'agit une chaîne de caractères, `$str` ou `$ch` peuvent être des bons choix.

Dans la majorité des cas, votre fonction retournera une valeur. Ceci est réalisé avec l'instruction `return` :

```
return $resultat;
```



L'instruction `return`

L'instruction `return` met fin à la fonction en renvoyant un résultat relatif à la donnée qui lui est passée en paramètre. Une fonction contenant des conditions peut tout à fait avoir différentes instructions `return`.

Une fois votre fonction déclarée dans votre script PHP, vous pouvez l'appeler de n'importe quel endroit et autant de fois que vous le voulez :

```
function double($n)
{
  $resultat = $n * 2;
  return $resultat;
}
```

```
$a = 2;
$b = double($a);
print("le double de $a est $b");
// affiche : "le double de 2 est 4"
```

Écrivons notre propre version de la fonction de calcul de puissance : `pow()`. Vous ne pouvez pas l'appeler `pow()` car il est interdit d'utiliser le nom d'une fonction déjà existante. Nommons-la `puissance` :

```
function puissance($n,$p)
{
  $resultat = $n;
  $i = 1;
  while ($i < $p)
  {
    $resultat *= $n;
    $i++;
  }
  return $resultat;
}

print(puissance(2,4)); // affiche comme prévu 16
```

Étudions maintenant un exemple dans lequel les fonctions rendent un véritable service. Le script suivant permet à un comptable de calculer le bonus de deux commerciaux selon une formule assez complexe :

```
si le CA est supérieur à 300 :
  bonus = (CA + 100) / 8.4
sinon
  bonus = (CA + 80) / 7.6
```

```
$ca_marcel = 350;
$ca_julien = 150;

if ($ca_marcel > 300)
  $bonus_marcel = ($ca_marcel + 100) / 8.4;
else
  $bonus_marcel = ($ca_marcel + 80) / 7.6;

print("Bonus de Marcel : ".$bonus_marcel."<br/>");

if ($ca_julien > 300)
  $bonus_julien = ($ca_julien + 100) / 8.4;
else
  $bonus_julien = ($ca_julien + 80) / 7.6;

print("Bonus de Julien : ".$bonus_julien."<br/>");
```

Première remarque à la vue de ce code : vous avez écrit deux fois le même extrait de code :

```
if ($ca > 300)
  $bonus = ($ca + 100) / 8.4;
else
  $bonus = ($ca + 80) / 7.6;
```

C'est doublement regrettable ! Tout d'abord, il est extrêmement fastidieux de taper du code (nous sommes tous d'accord sur ce point) et il est donc inutile de perdre son temps à écrire plusieurs fois la même chose. Imaginez ensuite que vous vouliez modifier le taux 8,4 par 8,3. Il vous faudra non seulement le faire à différents endroits, mais, qui plus est, il conviendra de n'oublier aucune mise à jour. C'est précisément pour éviter ces difficultés qu'il est bon de créer la fonction `calc_bonus()` qui calcule le bonus d'un commercial en fonction de son CA :

```
function calc_bonus($prenom,$ca)
{
  if ($ca > 300)
    $bonus = ($ca + 100) / 8.4;
  else
    $bonus = ($ca + 80) / 7.6;
  print("Bonus de ".$prenom." : ".$bonus."<br/>");
  return $bonus;
}
```

Le code devient donc :

```
function calc_bonus($prenom,$ca)
{
  if ($ca > 300)
    $bonus = ($ca + 100) / 8.4;
  else
    $bonus = ($ca + 80) / 7.6;
  print("Bonus de ".$prenom." : ".$bonus."<br/>");
  return $bonus;
}

calc_bonus("marcel",350);
calc_bonus("julien",150);
```

Cette modification a aussi permis de rendre plus clair et plus lisible votre code, ce qui n'est pas négligeable.



Commentaires

Les commentaires sont autorisés au sein même de la définition d'une fonction.

Valeurs par défaut

Nous avons vu que la syntaxe pour déclarer une fonction est la suivante :

```
function nom_fonction($param1,$parma2)
{
  bloc_de_code;
}
```

Il est possible de donner des valeurs par défaut aux paramètres de la fonction en la déclarant ainsi :

```
function nom_fonction($param1 = "valeur_par_défaut")
{
  bloc_de_code;
}
```

Écrivez une fonction `affiche_retour()` et appelez-la de deux façons différentes :

```
<?php

function affiche_retour($url = "/")
{
  echo "<a href=$url>retour</a><br/>";
}

// version 1 : nous utilisons le paramètre par défaut
echo "merci de votre visite<br/>";
affiche_retour();

//version 2 : nous passons la valeur du paramètre
echo "merci de votre visite<br/>";
affiche_retour("/");

?>
```

Attention, si la fonction possède plusieurs paramètres, seul le dernier pourra prendre une valeur par défaut !

```
<?php

function affiche_retour($url,$nomlien = "retour")
{
  echo "<a href=$url>retour</a><br/>";
}

//version 2 : valeur par défaut pour le deuxième paramètre
echo "merci de votre visite<br/>";
affiche_retour("/");
```

```
//version 2 : nous passons les valeurs des 2 paramètres
echo "merci de votre visite<br/>";
affiche_retour("/","back");

?>
```

Récursivité

La factorielle de la valeur `5` se calcule ainsi :

```
facto(5) = 5 * 4 * 3 * 2 * 1
```

Une définition possible de `facto(n)` est la suivante :

```
facto(n) = n * n-1 * n-2 * ... * 3 * 2 * 1
```

Il est aussi possible de donner une définition récursive de la fonction factorielle :

```
facto(n) = n * facto(n-1) avec facto(1) = 1
```

En effet, prenez `n = 5` et développez :

```
facto(5) = 5 * facto(4)
facto(4) = 4 * facto(3)
facto(3) = 3 * facto(2)
facto(2) = 2 * facto(1)
facto(1) = 1 (d'après la définition)
```

Si vous remontez maintenant avec tous vos résultats intermédiaires, vous obtenez :

```
facto(2) = 2 * 1
facto(3) = 3 * 2 * 1
facto(4) = 4 * 3 * 2 * 1
facto(5) = 5 * 4 * 3 * 2 * 1
```

Vous parvenez donc bien au résultat escompté. Cette technique de programmation, où une fonction se rappelle elle-même, s'appelle la « récursivité ». Il est possible en PHP d'utiliser cette technique.

Écrivez la fonction factorielle tout d'abord classiquement :

```
function facto($n)
{
  for ($resultat = 1; $n > 1; $n--)
  {
    $resultat *= $n;
  }
  return $resultat;
}

print(facto(5));
```



Boucle `for`

Vous pouvez vous apercevoir que l'indice `$n` diminue au sein de la boucle : `$n––`.

Écrivez maintenant la version récursive de la même manière que la formule récursive vue précédemment :

```
function facto($n)
{
  if ($n == 1) return 1;
  else return $n * facto($n-1);
}

print(facto(5));
```

En appelant `facto(5)`, la fonction va se rappeler elle-même jusqu'à `facto(1)`, où elle va retourner la valeur `1`. À partir de là, elle va faire le chemin inverse et remonter avec les valeurs intermédiaires.

Modifiez un peu le programme afin qu'il affiche les étapes intermédiaires :

```
<?php

function facto($n)
{
  print "facto de $n<br/>";
  if ($n == 1)
  {
    print "<br/>facto 1 = 1<br/><br/>";
    return 1;
  }
  else
  {
    $r = $n * facto($n-1);
    print "> $r<br/>";
    return $r;
  }
}

$n = 5;

print("<hr>factorielle de $n = " . facto($n));

?>
```

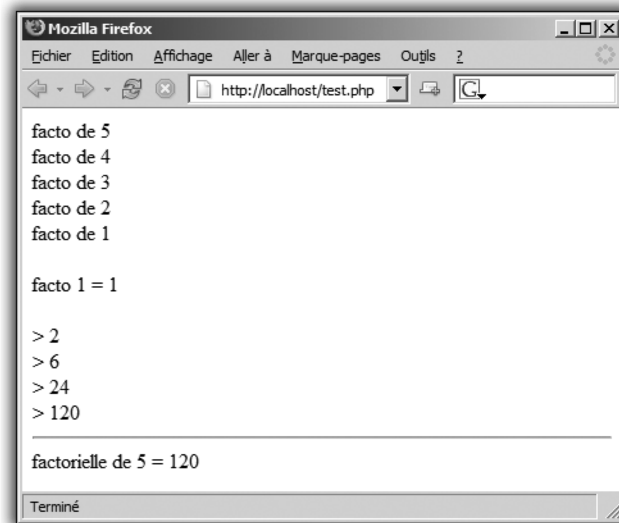


Figure 3.18 : *Factorielle*



Le parcours d'un répertoire écrit de façon récursive est proposé dans le chapitre consacré aux fonctions PHP dans la présentation de `opendir()`.

Cette technique est loin d'être indispensable en programmation. Tout code récursif peut être exprimé de manière itérative avec de simples boucles `for`. Néanmoins, dans certains cas, une version récursive peut se révéler beaucoup plus simple à coder.

Variables globales

Observez l'exemple suivant :

```
function aff()
{
  $i = $i + 1;
  print($i);
}

for ($i = 1; $i <= 5; $i++)
{
  print("[$i] ");
  aff();
  print("<br/>");
}
```

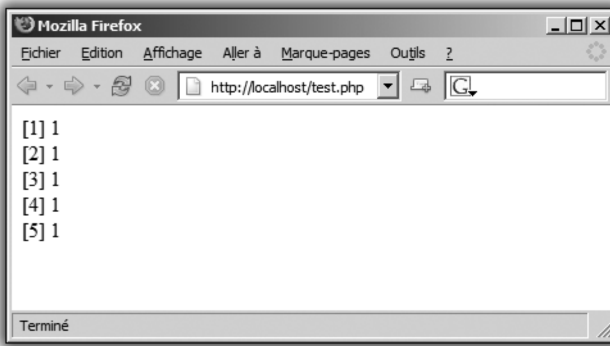


Figure 3.19 : *Fonction aff()*

L'événement qu'il faut noter, dans cet exemple, est que la variable `$i` de la fonction, d'une part, et celle du script, d'autre part, ne semblent pas être reliées entre elles, alors qu'elles ont le même nom. Quand vous êtes à la deuxième itération de votre boucle `for` et que `$i` vaut `2`, vous pouvez supposer que la fonction `aff()` affiche `3` (`$i + 1`) à l'écran et non `1`. Ce n'est pas le cas et ce comportement est tout à fait normal. Les variables incluses dans une fonction sont locales par rapport à la fonction et n'ont rien à voir avec les éventuelles variables de même nom présentes dans le corps du script. Il est cependant possible, depuis une fonction, d'avoir accès aux variables du script en déclarant dans la fonction les variables dont vous avez besoin en tant que « globales ».

```
global $var1, $var2 ....;
```

Modifiez la fonction `aff()` afin de lui faire utiliser la variable `$i` contenue dans le script :

```
<?php

function aff()
{
  global $i;
  $i = $i + 1;
  print($i);
}

for ($i = 1; $i <= 5; $i++)
{
  print("[$i] ");
  aff();
  print("<br/>");
}

?>
```

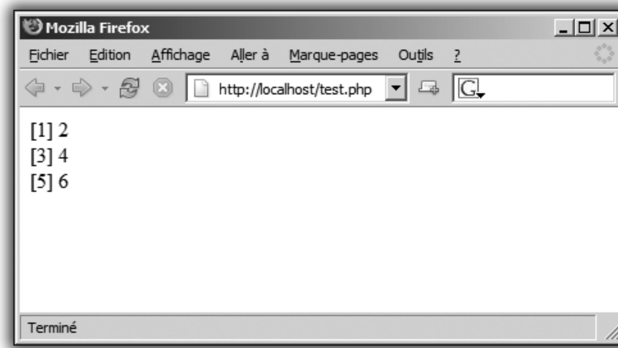


Figure 3.20 : *Exemple de fonction utilisant une variable globale*

Le résultat n'a plus rien à voir. Déroulez les deux premières itérations du script étape par étape…

- Première itération :

  ```
  entrée dans la boucle for : $i est initialisé à 1
  affichage de $i : 1
  appel de la fonction aff()
  récupération la variable $i du script qui vaut 1
  incrémentation de 1, $i vaut maintenant 2
  affiche de $i : 2
  ```

- Deuxième itération :

  ```
  incrément la valeur de $i qui vaut maintenant 3
  affichage de $i : 3
  appel de la fonction aff()
  récupération de $i : 3
  $i passe à 4
  affichage de $i : 4, etc.
  ```

Voyez maintenant si la modification suivante vous surprend :

```
<?php

function aff($i)
{
  $i = $i + 1;
```

```
  print($i);
}

for ($i = 1; $i <= 5; $i++)
{
  print("[$i] ");
  aff($i);
  print("<br/>");
}

?>
```

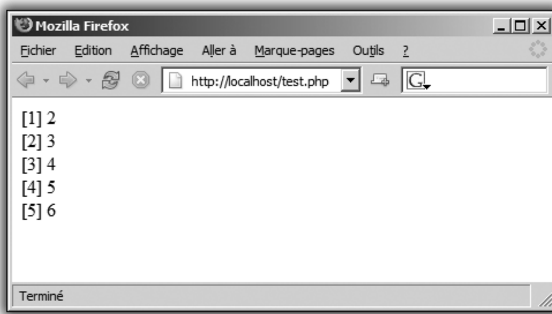


Figure 3.21 : *Résultat*

Dans ce cas, `aff` utilise bien la valeur de la variable `$i` du script, mais ne modifie pas sa valeur au sein du script. Rien de plus logique car `aff` reçoit en paramètre la valeur de `$i`, qu'elle stocke dans une variable locale : `$i`. Il ne s'agit en aucun cas d'une variable globale. Il faut donc retenir que les paramètres des fonctions sont des variables locales à la fonction comme les autres.

Inclusion de fichier

Un script PHP peut à tout moment faire appel à d'autres scripts par l'intermédiaire de la fonction `include()`. Variables et fonctions sont partagées par l'ensemble des scripts inclus. Cette technique est extrêmement intéressante pour charger un fichier de configuration ou pour construire votre interface graphique en y incluant par exemple systématiquement un en-tête et un pied de page. La modification d'un de ces fichiers inclus est tout de suite répercutée dans l'ensemble du site.

Cette fonction prend en argument le chemin du script à inclure qui peut être transmis de façon relative ou absolue.

Un chemin absolu correspond au chemin depuis la racine du serveur. Pour inclure le script *entete.php* présent dans le répertoire *www/interface* de Wamp, le chemin absolu correspond à `"c:/wamp/www/interface/entete.php"`.

Un chemin relatif correspond quant à lui au chemin depuis celui du script appelé :

- `include("entete.php")` inclut un script présent dans le même répertoire que le script appelé ;
- `include("interface/entete.php")` inclut un script présent dans le répertoire interface qui est lui-même présent dans le répertoire contenant le script appelé ;
- `include("../entete.php")` permet d'appeler un script situé dans le répertoire parent de celui du script appelé.



Inclusion d'inclusion

Si un « fichier inclus » inclut lui-même un autre fichier, le répertoire de référence pour les chemins reste quoi qu'il arrive celui du script appelé.

Le chemin relatif est généralement préférable car non dépendant de l'environnement de développement. Une application développée en relatif sous Windows peut tout à fait fonctionner dans un environnement Linux.

L'exemple suivant met en œuvre la fonction `include()` pour construire une structure de page composée d'un en-tête, d'un contenu et d'un pied de page. Le titre du site est contenu dans un fichier de configuration.

Listing 3-28 : entete.php

```
<html>

<title>
<?php
print($titre);
?>
</title>

<body>

<h1>
<?php
print($titre);
?>
```

```
</h1>

<hr/>
```

Listing 3-29 : piedpage.php

```
<hr/>
<center>
<i>contact - copyright - à propos de
<?php
print($titre);
?>
</i>
</center>

</body>

</html>
```

Listing 3-30 : config.php

```
<?php
$titre = "Cool Site";
?>
```

Listing 3-31 : test.php

```
<?php

include("interface/config.php");

include("interface/entete.php");
print("Bienvenue");
include("interface/piedpage.php");

?>
```



Figure 3.22 : *Modèle de page avec en-tête et pied de page*

La fonction `include()` peut également retourner une valeur si le code inclus utilise la fonction `return()`. Cette façon d'opérer est très utile pour savoir si l'exécution du code inclus s'est bien déroulée.

Listing 3-32 : Vérification de la valeur retounée par la fonction include()

```
if (include("script.php") == false) {
  die("Erreur");
}
```



Extension du script inclus

PHP interprète tous les scripts inclus quelle que soit leur extension. Il est ainsi courrant d'utiliser l'extension `.inc` ou `.inc.php` pour nommer les fichiers inclus et les distinguer des scripts accessibles « directement ». Ce n'est cependant ni une norme ni une obligation.

PHP propose trois autres fonctions très proches d'`include()` :

Tableau 3.4 : *Fonctions cousines de* `include()`

Nom de la fonction	Utilisation
`include_once()`	Le fichier n'est inclus que s'il ne l'a pas été auparavant.
`require()`	Cette fonction arrête le script si elle n'est pas parvenue à inclure le fichier passé en argument. La fonction `include()` se contente quant à elle d'afficher un avertissement et poursuit l'exécution du script.
`require_once()`	Le fichier n'est inclus que s'il ne l'a pas été auparavant.

Avec l'expérience, vous constaterez que les fichiers inclus sont souvent regroupés dans une même arborescence. Il peut donc apparaître superflu de spécifier systématiquement le chemin complet du script à inclure. PHP a prévu cela et permet de spécifier un certain nombre de répertoires dans lesquels l'interpréteur essayera de trouver les fichiers à inclure. Cette directive, `include_path`, contient une liste de répertoires séparés par un `;` sous Windows et par un `:` sous Linux / Unix. Cette directive (comme un certain nombre d'autres) peut être modifiée soit directement dans le fichier de configuration `php.ini` soit à l'aide des fonctions `ini_get()` et `ini_set()` qui permettent de récupérer la valeur d'une

directive pour la première et de la modifier pour la seconde. L'exemple ci-dessus peut ainsi être modifié de la façon suivante :

Listing 3-33 : Modification de la directive include_path afin de simplifier les inclusions

```
<?php

$tmp = ini_get("include_path").";interface";
ini_set("include_path",$tmp);

include("config.php");

include("entete.php");
print("Bienvenue");
include("piedpage.php");

?>
```



Inclusion de fichiers distants

La fonction `include()` autorise l'inclusion de fichiers distants. L'instruction `include("http://www.google.com")` est par conséquent tout à fait valide. Veillez toutefois à contrôler que la directive de configuration `allow_url_fopen` soit bien à `On`.

3.8. Check-list

- PHP est un langage procédural.
- Les variables sont précédées d'un `$`.
- Un point-virgule doit être présent à la fin de chaque instruction.
- Un groupe d'instructions PHP doit être entouré des balises `<?php` et `?>`.
- PHP n'impose pas le typage de variables en début de programme. Les types sont toutefois présents et comptent, parmi les plus importants, les numériques et les chaînes de caractères.
- Les structures de contrôle permettent de restreindre l'exécution d'instructions à certains cas bien spécifiques.
- Les boucles permettent l'exécution répétitive de certaines instructions.
- Les fonctions permettent d'isoler des groupes d'instructions qui auraient été sinon répétés dans le code. En plus des fonctions internes à PHP, le développeur peut créer ses propres fonctions.

- Les fonctions peuvent avoir accès aux variables du programme à l'aide du mot-clé `global`.
- Comme les fonctions, les fichiers inclus permettent de mieux organiser votre code.

Chapitre 4

Les tableaux

Présentation 116
Parcours d'un tableau 121
Les fonctions 125
Les opérateurs sur les tableaux 136
Check-list 137

Les variables étudiées jusqu'à présent permettaient de ne stocker qu'une seule et même valeur, qu'il s'agisse d'une chaîne de caractères, d'un numérique ou d'un booléen.

PHP propose également un type spécial de donnée donnant la possibilité à une variable de contenir une liste d'éléments. Ces variables sont appelées tableaux et sont associées au type `array`. Les tableaux peuvent être assimilés à des classeurs disposant d'un nombre variable d'emplacements pour stocker de l'information. On parle, pour qualifier ces emplacements, de cellules d'un tableau.

Ce chapitre va nous permettre de nous familiariser avec ce type de variable en présentant les opérations de création et de manipulation de tableaux. Nous étudierons également les différents opérateurs ainsi qu'un certain nombre de fonctions spécifiques aux tableaux.

4.1. Présentation

Deux méthodes permettent de créer et d'initialiser un tableau.

La première consiste à utiliser la fonction `array()` en lui passant directement en argument la liste des éléments du tableau. La création d'un tableau `$couleur` contenant les éléments `"rouge"`, `"vert"`, `"bleu"` utilise la syntaxe suivante :

Listing 4-1 : Création d'un tableau $couleur

```
$couleur = array("rouge", "vert", "bleu");
```

L'autre méthode consiste à utiliser la fonction `array()` sans argument afin de déclarer la variable en tant que tableau puis à lui ajouter des éléments. L'ajout d'une cellule utilise la syntaxe `$couleur[]`.

Listing 4-2 : Création du tableau $couleur en lui ajoutant 3 cellules

```
$couleur = array();
$couleur[] = "rouge";
$couleur[] = "vert";
$couleur[] = "bleu";
```

Les deux méthodes peuvent être cumulées. Dans l'exemple suivant, le tableau `$couleur` est créé avec deux cellules initiales puis se voit complété par trois cellules complémentaires.

Listing 4-3 : $couleur contient cinq cellules

```
$couleur = array("rouge","vert");
$couleur[] = "bleu";
$couleur[] = "jaune";
$couleur[] = "orange";
```

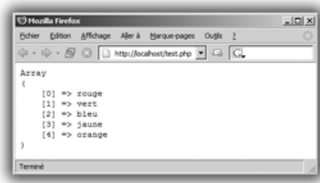


Figure 4.1 :
Tableau contenant cinq cellules

Maintenant que nous sommes en mesure de créer des tableaux, l'étape suivante consiste à pouvoir faire référence à une cellule d'un tableau en particulier. Nous allons pour cela présenter les deux grandes familles de tableaux : les tableaux scalaires et les tableaux associatifs.

Les tableaux scalaires

Dans le cadre d'un tableau scalaire, chaque élément du tableau est accessible par son numéro unique d'enregistrement, aussi appelé index ou indice.

Par convention, le premier élément d'un tableau a pour index `0`. `$couleur[0]` correspond donc à la première valeur du tableau, c'est-à-dire à `"rouge"`.

Listing 4-4 : Accéder à un élément d'un tableau

```
$couleur = array("rouge", "vert", "bleu");
print("deuxième élément du tableau : $couleur[1]");
// affiche : "deuxième élément du tableau : vert"
```

La modification d'un élément de tableau se fait en affectant une nouvelle valeur directement à l'élément en question :

Listing 4-5 : Modifier une valeur d'un tableau

```
$couleur = array("rouge", "vert", "bleu");

// modification du premier élément du tableau couleur :
// il ne contiendra plus "rouge" mais "jaune"
$couleur[0] = "jaune";
```

L'association d'une valeur à une cellule dont l'indice n'existe pas permet de créer cette cellule :

```
$couleur = array("rouge", "vert", "bleu");

// ajoute la couleur orange au tableau couleur qui
//contenait 3 valeurs
$couleur[3] = "orange";
print("$couleur[3]") ; // affiche le quatrième élément du
// tableau : orange
```



Ordre des indices

PHP ne vous impose pas de suivre un ordre rigoureux dans le choix des indices du tableau. L'exemple suivant est par exemple tout à fait valide :

```
$couleur = array();
$couleur[0] = "bleu";
$couleur[2] = "jaune";
$couleur[4] = "orange";
```

À charge pour vous de trouver une logique à tout cela !

Les tableaux associatifs

Dans le cadre d'un tableau associatif, chaque cellule du tableau est identifiée non plus par un indice numérique mais par une clé de type chaîne de caractères. Cette clé, par nature, doit être unique afin de permettre de sélectionner chaque élément du tableau.

La création d'un tableau associatif utilise la syntaxe suivante :

```
array(cle1 => valeur1, cle2 => valeur2)
```

La création d'un tableau `$personne` dont les caractéristiques seraient le nom, le prénom et l'âge peut être réalisée de la manière suivante :

```
$personne = array("nom"=>"Dupont","prenom"=>"Paul","age"=>23);
```

La syntaxe `$personne["nom"]` permet quant à elle d'accéder à la valeur de la cellule identifiée par la clé `nom`.

L'assignation d'une valeur à une cellule fonctionne comme pour les tableaux scalaires, en remplaçant l'indice par la clé : `$personne["age"] = 34`.

Si la clé n’existe pas, une nouvelle cellule est alors automatiquement créée.

Listing 4-6 : Le tableau contient désormais quatre cellules

```
$tab = array("nom"=>"Dupont","prenom"=>"Paul","age"=>23);
$tab["ville"] = "Paris";
```



Majuscules et minuscules pour les clés

Les clés sont sensibles à la casse. Ainsi `$personne['nom']` et `$personne['Nom']` feront référence à deux cellules distinctes.

La définition d’un tableau sur plusieurs lignes favorise souvent la lecture et la compréhension des sources.

Listing 4-7 : Définition d’un tableau sur plusieurs lignes

```
$tab = array("nom" => "Dupont",
             "prenom" => "Paul",
             "age" => 23);
```

Les tableaux multidimensionnels

Les différents tableaux présentés précédemment sont des tableaux à une dimension. Un index seul (ou une clé seule) donne la possibilité d’accéder à n’importe quel élément du tableau.

PHP vous permet également de créer des tableaux multidimensionnels. Un tableau de dimension `2` par exemple peut être envisagé comme un échiquier. Deux informations sont nécessaires pour accéder à une case donnée du tableau : l’abscisse et l’ordonnée.

Pour accéder à l’élément d’abscisse `3` et d’ordonnée `5` du tableau `$echiquier`, la syntaxe à utiliser est la suivante `$echiquier[3][5]`.

Des tableaux multidimensionnels scalaires et associatifs peuvent également être créés. Illustrons cela à l’aide d’un exemple et créons le tableau `$tab` :

Listing 4-8 : Création d’un tableau à 2 dimensions

```
$tab = array(
  0 => array("prenom"=>"pe", "nom"=>"wood","age"=>12),
  1 => array("prenom"=>"fc", "nom"=>"bosque","age"=>11)
  );
```

Ce tableau contient deux individus, chacun disposant d'un prénom, d'un nom et d'un âge.

Il est important ici de comprendre que chaque élément du tableau est lui-même composé d'un autre tableau. Le schéma suivant permet de mieux visualiser la structure interne de `$tab` :

0	1
PRENOM	PRENOM
pe	fc
NOM	NOM
wood	bosque
AGE	AGE
12	11

Figure 4.2 :
Tableau multidimensionnel

Pour augmenter `$tab` d'un nouvel individu, nous lui ajoutons une cellule qui, en l'occurrence, est elle-même un tableau :

Listing 4-9 : Ajout d'une cellule à $tab

```
$tab[] = array("prenom"=>"jp", "nom"=>"siob", "age"=>13);
```



Figure 4.3 :
État du tableau multidimensionnel après l'ajout d'un nouvel individu

Pour accéder au prénom du troisième individu, écrivez : `$tab[2]["nom"]`. N'oubliez pas en effet que le premier élément a pour index `0`.



Affichage complexe

Faire référence à la cellule d'un tableau scalaire au sein d'une chaîne de caractères est tout à fait autorisé :

```
print("premier élément : $tab[0]");
```

Il en va tout autrement pour les tableaux scalaires et les tableaux multidimensionnels qui imposent une syntaxe dite « complexe ». Deux accolades viennent alors entourer la référence à la cellule :

```
print("élément nom : {$tab['nom']}");
print("élément nom : {$tab[2]["nom"]}");
```

La solution alternative consistant à utiliser l'opérateur de concaténation est bien évidemment toujours autorisée :

```
print("élément nom : " . $tab[2]['nom']);
```

4.2. Parcours d'un tableau

Diverses méthodes permettent de parcourir l'ensemble des valeurs d'un tableau.

Boucle foreach

PHP propose une catégorie de boucle `for` spécialement dédiée aux tableaux : la boucle `foreach`. Une boucle `foreach` pourrait être traduite de la manière suivante :

```
POUR chaque élément contenu dans le tableau FAIRE
```

La syntaxe suivante permet d'afficher l'ensemble des éléments d'un tableau :

Listing 4-10 : Affichage des éléments d'un tableau scalaire

```
$tab = array ("a", "b", "c", "d");
foreach ($tab as $val) {
  print("$val<br/>");
}
```

À chaque itération de la boucle, `foreach` associe la valeur de l'élément du tableau en cours à la variable `$val` puis déplace son pointeur interne sur l'élément suivant. Le nom de la variable est bien évidemment libre.

S’il s’agit d’un tableau associatif, la syntaxe est légèrement différente afin de permettre la récupération de la clé associée.

Listing 4-11 : Affichage des éléments d’un tableau associatif

```
$tab = array("prenom" => "paul", "nom" => "dupont");
foreach ($tab as $cle => $valeur) {
  print("$cle = $valeur<br/>");
}
```

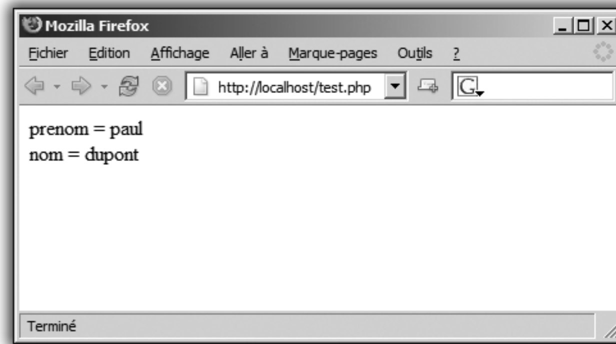


Figure 4.4 : *Boucle foreach*



Écriture clé/valeur pour un tableau scalaire

L’écriture `foreach ($tab as $cle => $valeur)` est également possible pour les tableaux scalaires. La variable `$cle` contiendra alors l’indice (numérique) de l’élément.

L’utilisation d’une boucle `while` pour balayer un tableau est assez courante (surtout parmi les développeurs issus du langage C). L’idée est ici de parcourir le tableau tant qu’il contient un élément :

Listing 4-12 : Parcours d’un tableau à l’aide de la boucle while

```
$tab = array("a", "b", "c", "d");
$i = 0;
while ($tab[$i]) {
  print($tab[$i]."<br/>");
  $i++;
}
```

Cette méthode trouve cependant sa limite avec un tableau `$tab` qui contiendrait les valeurs suivantes : `"a", "b", false, "c"`.

Le `while` s'arrêtera en effet à la troisième cellule lorsque `$tab[$i]` contiendra la valeur `false`. Une boucle `foreach` en revanche afficherait bien les quatre éléments.



`break` **et** `continue`

Les instructions `break` et `continue` peuvent également être utilisées dans le cadre d'un `foreach`.

Utilisation du pointeur interne

En plus des éléments qu'il contient, un tableau dispose également d'un pointeur interne sur l'élément courant. Le pointeur est placé par défaut sur le premier élément du tableau et peut être déplacé à l'aide d'une multitude de fonctions.

Tableau 4.1 : *Fonctions permettant de manipuler le pointeur interne d'un tableau*

Fonction	Rôle
`reset()`	Place le pointeur au début du tableau et retourne la valeur du premier élément
`next()`	Déplace le pointeur sur l'élément suivant et retourne sa valeur
`prev()`	Déplace le pointeur sur l'élément précédent et retourne sa valeur
`key()`	Retourne la clé de l'élément courant
`each()`	Retourne un tableau contenant la paire clé/valeur de l'élément courant
`current()`	Retourne la valeur de l'élément courant
`end()`	Place le pointeur sur le dernier élément de la liste et retourne sa valeur

Listing 4-13 : Déplacement du pointeur interne d'un tableau

```
$tab = array("rouge","vert","bleu","jaune");
echo end($tab)."<br/>";
echo prev($tab)."<br/>";
echo key($tab)."<br/>";
echo reset($tab)."<br/>";
echo current($tab)."<br/>";
print_r(each($tab))."<br/>";
```

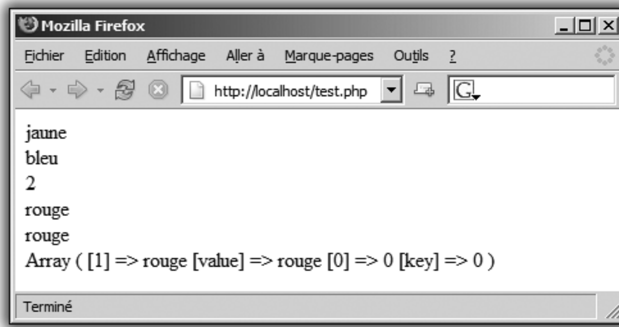


Figure 4.5 : *Exemple de déplacement de pointeur*



Fonction `end()`

Cette fonction peut retourner la valeur `null` si le pointeur interne est déjà au bout du tableau ou si le dernier élément contient lui-même la valeur `null`. La boucle `foreach` reste donc le meilleur moyen de parcourir un tableau de façon sûre.

Utilisation des références

L'assignation d'une variable à une autre variable ne signifie en aucun cas que les deux variables sont identiques mais bien que leur valeur à un instant `t` sont les mêmes.

Listing 4-14 : Assignation simple, les 2 variables sont indépendantes

```
$a = 2;
$b = $a; // $a et $b contiennent 2
$a = 1; // $a contient 1 et $b contient toujours 2
```

PHP dispose toutefois d'une syntaxe permettant de lier de façon forte deux variables. Il s'agit de la notion de référence. L'opérateur `&` devant une variable permet de donner accès non pas à sa valeur mais à une référence à elle-même. Toute modification d'une variable modifiera alors l'autre.

Listing 4-15 : $b est un alias de $a

```
$a = 2;
$b = &$a; // $b n'est qu'une référence à $a
$a = 1; // $a contient 1 et $b contient également 1
```

Cette syntaxe est particulièrement intéressante dans le cadre d'un `foreach` pour pouvoir modifier une à une toutes les valeurs d'un tableau. Chaque itération de la boucle donne accès non pas à la valeur de l'élément mais à une référence à cet élément permettant de modifier sa valeur.

Listing 4-16 : Chaque itération donne accès à une référence

```
$tab = array("bleu", "blanc", "rouge");
foreach ($tab as &$value) {
  $value = "{".$value."}";
}
```

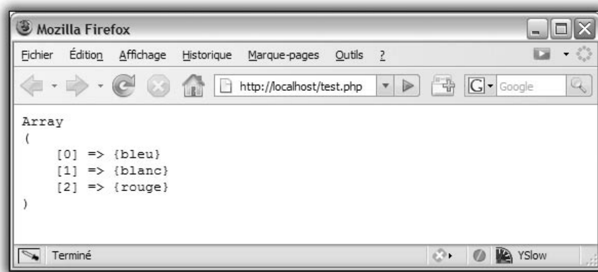


Figure 4.6 : *Tous les éléments du tableau ont bien été modifiés*

4.3. Les fonctions

Plus de 70 fonctions consacrées aux tableaux sont mises à la disposition du programmeur PHP. Ces fonctions, aussi souples que puissantes, sont sans conteste une des grandes forces de ce langage.



Une liste plus complète de fonctions de manipulations de tableaux est proposée à la fin de cet ouvrage.

Suppression d'une cellule

La fonction `unset()` utilisée pour supprimer une variable sert également à supprimer des cellules d'un tableau.

Listing 4-17 : Suppression de la deuxième cellule du tableau

```
$tab = array("a","b","c");
unset($tab[1]);
```

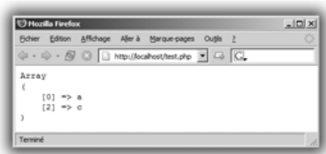


Figure 4.7 :
État du tableau après la suppression

La logique est exactement la même pour des tableaux multidimensionnels.

Listing 4-18 : Suppression du prénom associé à Mr Wood

```
$tab = array(
  0 => array("prenom"=>"pe", "nom"=>"wood","age"=>12),
  1 => array("prenom"=>"fc", "nom"=>"bosque","age"=>11)
  );
unset($tab[0]['prenom']);
```

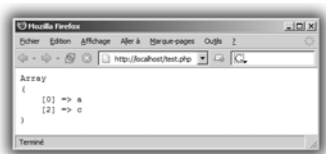


Figure 4.8 :
Seul le prénom du premier individu a été supprimé

Affichage d'un tableau

La fonction `print_r()` permet d'afficher un tableau d'une manière lisible. Créons un tableau complexe afin de visualiser l'intérêt de cette fonction :

Listing 4-19 : Affichage d'un tableau complexe

```
$tab = array(
  0 => array("a","b",array("d","e","f")),
  1 => "i",
  2 => array("j","k"=>array("l"=>array("m","o"),"p"))
  );
print_r($tab);
```

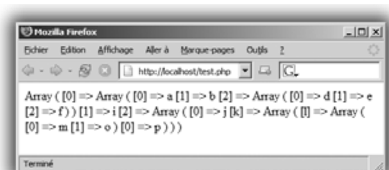


Figure 4.9 :
Utilisation de print_r() pour afficher le contenu de $tab

Nous constatons que cet affichage révèle le contenu du tableau sans faciliter réellement la lecture du tableau. Notre erreur est ici de ne pas entourer le `print_r()` des balises `<pre>` et `</pre>`. Ces balises permettent en effet de conserver l'affichage des retours chariots et des espaces au sein d'un contenu HTML.

Listing 4-20 : Balises <pre> et </pre> autour de la fonction print_r()

```
$tab = array(
  0 => array("a","b",array("d","e","f")),
  1 => "i",
  2 => array("j","k"=>array("l"=>array("m","o"),"p"))
  );

print("<pre>");
print_r($tab);
print("</pre>");
```

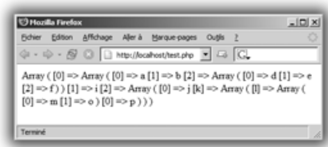


Figure 4.10 :
Amélioration de l'affichage du print_r() à l'aide des balises <pre> et </pre>

Taille d'un tableau

La taille d'un tableau peut être obtenue avec la fonction `count()`. Cette taille correspond au nombre d'éléments qu'un tableau contient, qu'il soit scalaire ou associatif.

```
$tab = array();
$n = count($tab);
print($n); // affiche : 0

$tab = array(1,2);
$n = count($tab);
print($n); // affiche : 2

$tab = array("a"=>"b");
$n = count($tab);
print($n); // affiche : 1

$tab = array("a"=>array(1,2,3),"b"=>array(4,5,6));
$n = count($tab);
print($n); // affiche : 2
```

La fonction `count()` peut également retourner la taille d'un tableau multidimensionnel si la valeur 1 est passée en second paramètre.

```
$tab = array("a"=>"b");
$n = count($tab,1);
print($n); // affiche : 1

$tab = array("a"=>array(1,2,3),"b"=>array(4,5,6));
$n = count($tab,1);
print($n); // affiche : 8
```

Le dernier exemple affiche 8 car il s'agit d'un tableau contenant 2 cellules qui elles-mêmes contiennent chacune trois cellules : 2+3+3 = 8.



Fonction `sizeof()`

La fonction `sizeof()` peut également être utilisée pour obtenir la taille d'un tableau. Cette fonction ne correspond cependant qu'à un alias de la fonction `count()` qui est, de ce fait, recommandée.

Conversion chaînes / tableaux

Une liste d'éléments est régulièrement représentée en informatique par une chaîne de caractères contenant ces mêmes éléments séparés par une virgule, un point-virgule, une tabulation ou tout autre caractère de séparation.

Listing 4-21 : Variable contenant une liste de prénoms

```
$liste = "patrick,pascal,veronique,benedicte,gonzague,olivier";
```

Ce type de représentation est extrêmement pratique pour stocker une liste dans un fichier de configuration ou lors d'une transmission de donnée mais se révèle très peu manipulable au sein d'un script PHP. Dans une telle situation, la représentation en tableau est largement préférable. La fonction `explode()` permet de convertir une liste représentée par une chaîne de caractères en un tableau. Le premier argument de la fonction correspond au caractère de séparation et le second à la chaîne elle-même. L'élément de séparation est souvent constitué par un caractère unique mais peut également correspondre à une chaîne de caractères telle que : `",,"`, `"##"`, etc.

Listing 4-22 : Conversion d'une chaîne en un tableau

```
$liste = "patrick,pascal,veronique,benedicte,gonzague,olivier";
print($liste);

$tab = explode(",",$liste);
print("<pre>");
print_r($tab);
print("</pre>");
```



Figure 4.11 :
Représentation d'une liste sous la forme d'une chaîne puis d'un tableau

La situation consistant à devoir assigner à différentes variables les valeurs d'un tableau retourné par `explode()` est assez courante.

Listing 4-23 : Association d'éléments d'un tableau à des variables

```
$date = "2004/10/23";
$tab = explode("/", $date);
$annee = $tab[0];
$mois = $tab[1];
$jour = $tab[2];
```

PHP permet de réduire cette syntaxe avec la fonction `list()`. Cette dernière se charge d'assigner les valeurs d'un tableau à plusieurs variables en une seule ligne.

Listing 4-24 : Assignation de 3 variables en une seule ligne

```
$date = "2004/10/23";
list($annee, $mois, $jour) = explode("/", $date);
```



Abus de langage

`list()` n'est pas véritablement une fonction; il s'agit plus exactement d'un élément constitutif du langage PHP.

L'opération inverse d'`explode()` permettant de passer d'un tableau à une chaîne est réalisée par la fonction `implode()` dont les deux arguments sont le caractère d'union et le tableau.

Listing 4-25 : Conversion d'un tableau en chaîne de caractères

```
$tab = array("patrick","pascal","veronique","benedicte",
✂ "gonzague","olivier");
print("<pre>");
print_r($tab);
print("</pre>");

$liste = implode(" ; ",$tab);
print($liste);
```



Figure 4.12 :
Trois caractères sont ici utilisés pour unir les éléments du tableau au sein de la chaîne

Adjonction, soustraction d'éléments

Certaines fonctions permettent d'ajouter ou d'enlever des éléments en début ou fin de tableau.

Tableau 4.2 : *Fonctions permettant d'ajouter et d'enlever des éléments d'un tableau*

Nom de la fonction	Rôle
`array_shift()`	Retourne le premier élément du tableau et le supprime
`array_pop()`	Retourne le dernier élément du tableau et le supprime
`array_unshift()`	Ajoute un ou plusieurs éléments au début du tableau
`array_push()`	Ajoute un ou plusieurs éléments à la fin du tableau

Illustrons ces différentes fonctions à l'aide d'un exemple :

Listing 4-26 : Evolution du contenu d'un tableau

```
$tab = array("rouge","jaune","vert");

print("<pre>");
print("<b>Tableau initial :</b><br/>");
print_r($tab);

$elem = array_shift($tab);
print("<br/><b>Après array_shift() :</b><br/>");
print_r($tab);

$elem = array_pop($tab);
print("<br/><b>Après array_pop() :</b><br/>");
print_r($tab);

array_unshift($tab,"orange");
print("<br/><b>Après array_unshift() :</b><br/>");
print_r($tab);
```

```
array_push($tab,"marron","gris");
print("<br/><b>Après array_push() :</b><br/>");
print_r($tab);
print("</pre>");
```

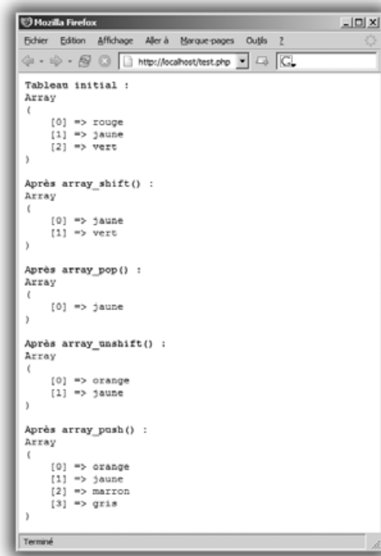


Figure 4.13 :
Affichage du contenu du tableau pour chaque étape

Tri

Les fonctions de tri sont nombreuses et permettent de prendre en compte la quasi-totalité des situations dans lesquelles vous pouvez vous trouver.

sort()

Cette fonction permet de trier un tableau en fonction de ses valeurs.

```
$tab = array("vert","bleu","jaune");
sort($tab);
```

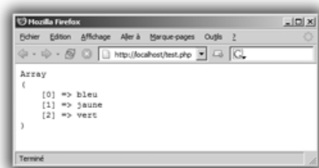


Figure 4.14 :
Tri du tableau $tab

Si le tableau est associatif, les clés sont alors supprimées.

```
$tab = array("nom"=>"dupont","age"=>"21","prenom"=>"alexandre");
sort($tab);
```

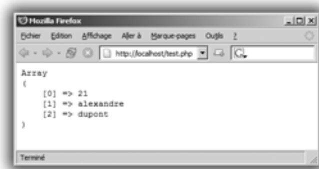


Figure 4.15 :
Le tableau est trié mais les clés ont disparu

asort()

Cette fonction permet de conserver l'association clé/valeur à l'issue du tri. Le tri porte sur les valeurs du tableau.

```
$tab = array("nom"=>"dupont","age"=>"21","prenom"=>"alexandre");
asort($tab);
```

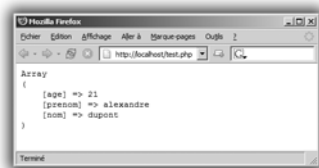


Figure 4.16 :
Les clés restent associées à leur valeur

ksort()

Le tri s'effectue cette fois sur les clés, tout en maintenant l'association avec les valeurs.

```
$tab = array("nom"=>"dupont","age"=>"21","prenom"=>"alexandre");
ksort($tab);
```

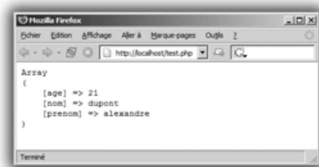


Figure 4.17 :
Tri sur les clés

natsort()

Les comparaisons sur les chaînes de caractères sont réalisées caractère par caractère. Ainsi la chaîne `"ab"` est plus « petite » que `"ac"` car le caractère `b` est plus petit que `c`. De la même manière, `"ab"` est plus petit que `"aaa"` car la première chaîne ne dispose pas de troisième caractère.

Ce mode de classement peut poser problème pour des valeurs contenant des chiffres. Dans un tel cas en effet, la chaîne `"fichier10.jpg"` devient inférieure à `"fichier2.jpg"`. La fonction `natsort()` est prévue pour ce type de situation et réalise un tri dit « naturel ».

```
$tab1 = $tab2 = array("fichier100.jpg","fichier2.jpg",
                        "fichier21.jpg","fichier1.jpg");

sort($tab1);
print("<b>SORT</b><hr/>");
print("<pre>");
print_r($tab1);
print("</pre>");

natsort($tab2);
print("<br/><b>NATSORT</b><hr/>");
print("<pre>");
print_r($tab2);
print("</pre>");
```

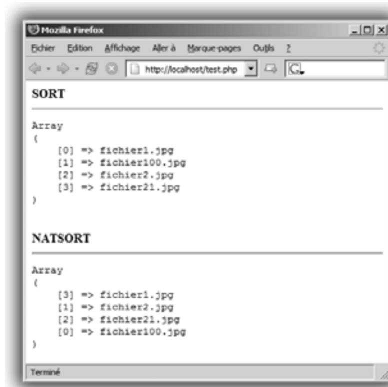


Figure 4.18 : *Différence entre un tri standard et un tri naturel*

Tri en ordre inverse

La plupart des fonctions présentées ci-dessus disposent d'une fonction « cousine » permettant de réaliser un tri en ordre inverse. Ces fonctions

contiennent un caractère `r` supplémentaire dans leur nom : `arsort()`, `krsort()`, `rsort()`.

Présence d'une valeur dans un tableau

La fonction `in_array()` permet de vérifier qu'une valeur donnée est bien présente au sein d'un tableau. Le premier argument de la fonction correspond à la valeur recherchée et le second au tableau dans lequel la recherche est réalisée.

Listing 4-27 : Vérification de la présence d'une valeur dans un tableau

```
$tab = array("thomas","henry");

if (in_array("thomas",$tab)) {
  print("Le prénom Thomas a été trouvé");
}
else {
  print("Le prénom Thomas n'a pas été trouvé");
}
```

Une fonction existe également pour vérifier la présence d'une clé au sein d'un tableau associatif `array_key_exists()`.

```
$tab = array("prenom"=>"thomas","nom"=>"henry");

if (array_key_exists("nom",$tab)) {
  print("Le nom est défini");
}
else {
  print("Le nom n'est pas défini");
}
```

Sérialisation

La sérialisation est une opération permettant de passer de la représentation binaire d'une donnée à une représentation textuelle. La fonction `serialize()` prend en argument un tableau et retourne une chaîne de caractères le représentant dans sa version sérialisée.

Listing 4-28 : Exemple d'utilisation de serialize()

```
$tableau = array("couleur" => array("rouge", "jaune"),
                 1 => 2);

print("<b>Affichage avec print_r()</b> :<br/>");
print("<pre>");
print_r($tableau);
```

```
print("</pre>");

print("<hr/>");

$tableau_s = serialize($tableau);
print("<b>Version sérialisée</b> :<br/><br/>".$tableau_s);
```

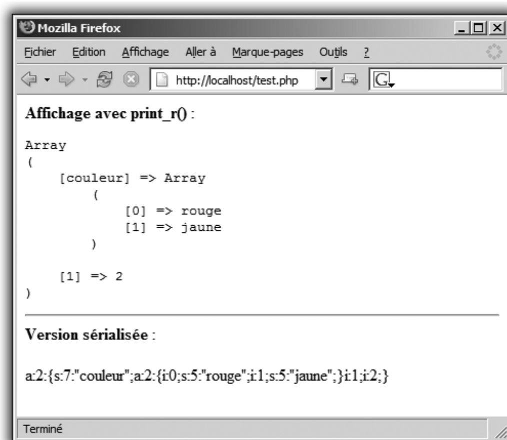


Figure 4.19 : *Deux représentations du tableau $tableau*

L'objectif, ici, n'est certainement pas d'analyser la syntaxe de la version sérialisée mais plutôt de comprendre que cette représentation peut tout à fait être placée au sein d'un fichier texte ou d'une base de données. Au lieu d'une présentation textuelle bancale de type CSV pour enregistrer le contenu d'un tableau, la représentation sérialisée possède le triple avantage d'être normée, optimisée et de disposer d'une fonction de conversion inverse : `unserialize()`.

L'exemple suivant illustre l'opération de désérialisation. La donnée contenue dans la variable `$str` pourrait tout à fait provenir du contenu d'un fichier ou d'une requête SQL sur une base de donnée.

Listing 4-29 : Illustration de la fonction unserialize()

```
$str = 'a:2:{i:0;s:1:"a";i:1;s:1:"b";}';

$tableau = unserialize($str);

print("<pre>");
print_r($tableau);
print("</pre>");
```

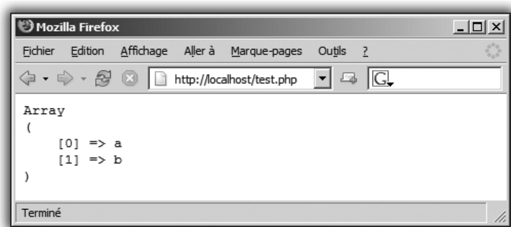


Figure 4.20 : *Opération de désérialisation*

4.4. Les opérateurs sur les tableaux

Bien qu'extrêmement puissants, les opérateurs sur les tableaux sont assez peu connus.

Tableau 4.3 : *Opérateurs sur les tableaux*

Opérateur	Résultat
`$a + $b`	Union de `$a` et de `$b`
`$a == $b`	Renvoie `true` si `$a` et `$b` sont composés des mêmes paires clé/valeur ; les tableaux sont alors dits égaux
`$a === $b`	Comme `==`, avec des vérifications supplémentaires sur l'ordre et le type des données ; les tableaux sont alors dits identiques
`$a != $b`	Renvoie `true` si `$a` et `$b` ne sont pas égaux
`$a <> $b`	Renvoie `true` si `$a` et `$b` ne sont pas égaux
`$a !== $b`	Renvoie `true` si `$a` et `$b` ne sont pas identiques

Illustrons ces opérateurs de quelques exemples et commençons par l'opérateur d'union.

L'opérateur d'union possède ceci de spécial que les clés qui seraient présentes dans `$a` et `$b` ne sont pas réécrites.

```
$a = array("couleur1"=>"rouge","couleur2"=>"vert");
$b = array("couleur1"=>"jaune","noir","vert");
print_r($a+$b);
```

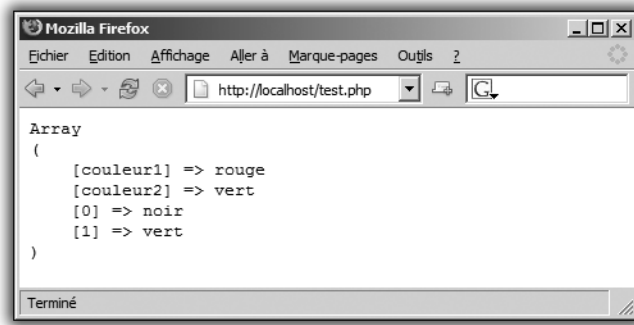


Figure 4.21 : *Union de $a et $b*

Arrêtons-nous maintenant sur les opérateurs de comparaisons afin de mieux comprendre les différences entre identiques et égaux.

Tableau 4.4 : *Différents tests de comparaison de tableaux*

Tableau 1	Tableau 2	Égaux ?	Identiques ?
`array(1,2)`	`array(1,2,3)`	NON	NON
`array(1,2)`	`array('1','2')`	OUI	NON
`array(1,2)`	`array(0=>1,1=>1)`	OUI	OUI
`array(1,2)`	`array(3=>1,4=>1)`	NON	NON
`array(1,2)`	`array(2,1)`	NON	NON

4.5. Check-list

- Le type associé aux tableaux est `array`.
- Les tableaux scalaires et associatifs correspondent aux deux grandes familles de tableaux.
- PHP est un des langages les plus souples pour la gestion des tableaux : il n'est pas demandé de préciser un nombre de cellules à la création, chaque cellule peut contenir des données de types différents.

Chapitre 5

Dates et heures

La notion de timestamp 140
Formatage d’une date 146
Contrôle de validité d’une date 152
Check-list 153

Les chapitres précédents ont permis de présenter les principaux types de données intégrés de façon native à PHP à savoir : les chaînes de caractères (`String`), les entiers (`Integer`), les nombres flottants (`Float`), les booléens (`Boolean`) ainsi que les tableaux (`Array`).

N'étant pas un langage purement objet, PHP n'intègre pas un type spécifique pour le traitement des dates. Le développeur dispose en revanche d'une quantité importante de fonctions permettant de réaliser les opérations les plus diverses.

5.1. La notion de timestamp

Un timestamp correspond à une représentation numérique d'une date. Il s'agit du nombre de secondes écoulées entre cette date et le premier janvier 1970.

La fonction `time()` retourne le timestamp actuel.

Listing 5-1 : Affichage du timestamp actuel

```
echo time()." secondes se sont écoulées depuis le 1er
✂ janvier 1970";
```

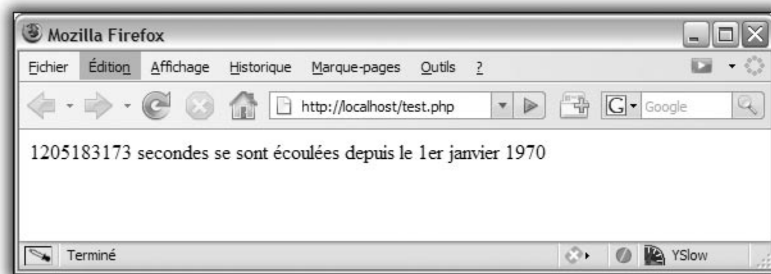


Figure 5.1 : *Affichage d'un timestamp*

L'actualisation de la page permet de constater que cette valeur est bien mise à jour toutes les secondes.



Timestamp et MySQL

Un timestamp MySQL est différent de son homologue PHP. MySQL propose cependant la méthode `UNIX_TIMESTAMP()` pour obtenir un timestamp « compatible » PHP.

Création d'un timestamp

L'obtention d'un timestamp autre que celui courant repose sur le fonction `mktime()`. Cette fonction prend 6 arguments au format numérique :

1 heure,
2 minute,
3 seconde,
4 mois,
5 jour,
6 année.

Le timestamp du 12 janvier 2008 peut être calculé avec l'instruction `mktime(0, 0, 0, 1, 12, 2008)`.

Une particularité bien pratique de `mktime()` réside dans sa faculté à accepter la valeur `0` pour le jour. La valeur retournée représente alors le timestamp du dernier jour du mois précédent. Cette ruse permet de ne pas avoir à se préoccuper du fait que le mois contienne 28, 29, 30 ou 31 jours.

Listing 5-2 : 2 écritures équivalentes pour le 31 janvier 2008

```
echo mktime(0, 0, 0, 2, 0, 2008)."<br/>";
echo mktime(0, 0, 0, 1, 31, 2008);
```

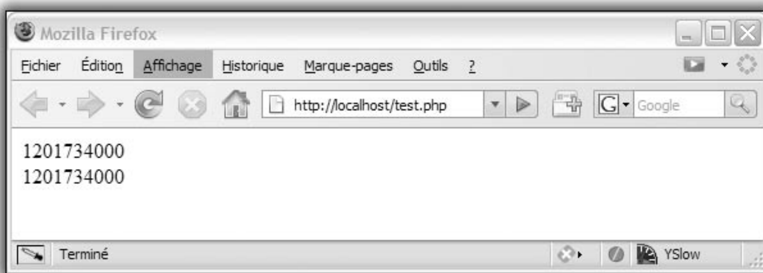


Figure 5.2 : *Les valeurs sont bien identiques*

Un timestamp correspondant à une quantité de secondes, l'opération consistant à ajouter le nombre de secondes dans une journée à un timestamp permet d'obtenir le timestamp du lendemain.

Listing 5-3 : 2 écritures équivalentes pour le 10 octobre 2008

```
echo mktime(0, 0, 0, 10, 10, 2008)."<br/>";
echo mktime(0, 0, 0, 10, 9, 2008) + (60 * 60 * 24);
```

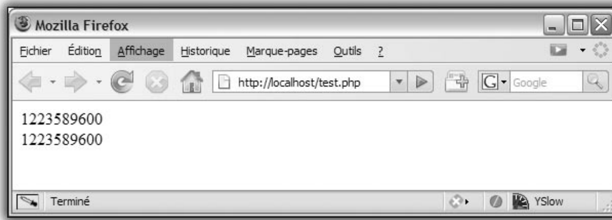


Figure 5.3 : *Le 10 octobre correspond bien au 9 octobre plus une journée*

Conversion

Qu'elle provienne d'un formulaire ou d'une base de données, un paramètre représentant une date sera le plus souvent transmis à un script sous la forme d'une chaîne de caractères. Le standard de notation le plus répandu correspond à la notation anglo-saxonne de la forme : `yyyy/mm/dd`. Le 12 janvier 2008 est noté `"2008/01/12"`.

Plutôt que de passer par un découpage fastidieux avec `explode()`, PHP propose la fonction magique `strtotime()`. Cette dernière prend en argument une chaîne de caractères représentant la date et retourne le timestamp équivalent.

Listing 5-4 : Utilisation de strtotime()

```
echo "timestamp correspondant au 12 Janvier 2008 :
✂ ".strtotime("2008/01/12");
```

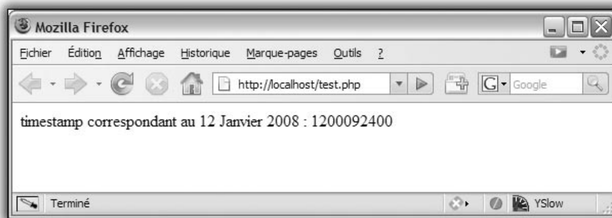


Figure 5.4 : *Utilisation de strtotime()*

La date peut être complétée avec une dimension horaire en utilisant la représentation suivante : `"yyyy/mm/dd hh:mm:ss"`. Minuit et une

minute le 12 janvier 2008 est représentée par la chaîne "2008/01/12 00:001:00". La fonction strtotime() accepte également ce format en argument.

Listing 5-5 : Présence d'un horaire

```
echo "timestamp correspondant à minuit et une minute le 12
✂ Janvier 2008 : ";
echo strtotime("2008/01/12 00:01:00");
```

Nous constatons que 60 secondes séparent bien le timestamp du "2008/12/01 00:00:00" et celui du "2008/12/01 00:01:00".

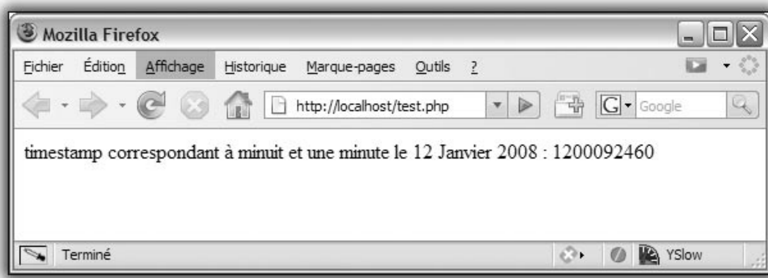


Figure 5.5 : *60 secondes de plus*

La fonction strtotime() accepte également des dates formatées en anglais courant.

Listing 5-6 : Deux écritures équivalentes

```
echo strtotime("2008/03/16")."<br/>";
echo strtotime("16 march 2008")."<br/>";
```

Plus intéressant encore, strtotime() peut interpréter certaines expressions complexes telles que :

- "+7 days" : timestamp dans 7 jours,
- "+1 week" : timestamp dans une semaine,
- "+6 days 24 hours" : timestamp dans 6 jours et 24 heures.

Listing 5-7 : Utilisation d'expressions complexes avec strtotime()

```
echo strtotime("+7 days")."<br/>";
echo strtotime("+1 week")."<br/>";
echo strtotime("+6 days 24 hours ")."<br/>";
```

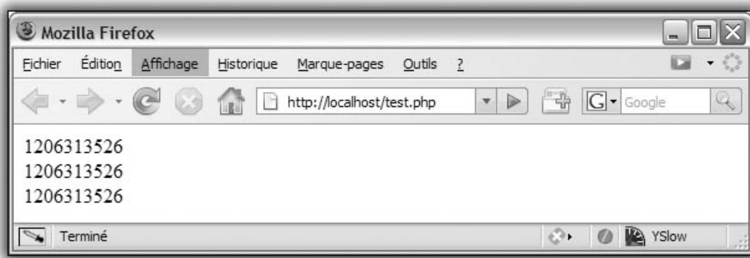


Figure 5.6 : *Les 3 expressions sont bien identiques*

Comparaison de dates

La représentation timestamp est essentielle pour pouvoir réaliser une comparaison entre deux dates. L'exemple suivant illustre le danger de comparer directement deux dates de la forme `"2008/2/7"` et `"2008/12/7"`.

Listing 5-8 : Comparaison directe entre deux dates au format chaîne de caractères

```
$date1 = "2008/3/7";
$date2 = "2008/12/7";
if ($date1 < $date2) {
  print("Le ".$date1." précéde le ".$date2);
}
else {
  print("Le ".$date1." succède au ".$date2);
}
```

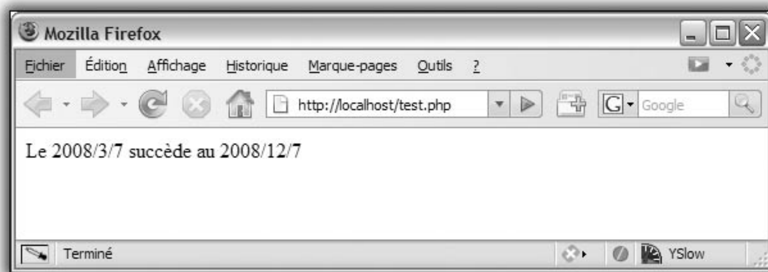


Figure 5.7 : *Résultat erroné*

PHP indique donc que le 7 mars 2008 succède au 7 décembre 2008 ! Le code ASCII du caractère `'3'` (`51`) étant supérieur à celui du caractère

'1' (49), ce résultat est par conséquent tout à fait cohérent du point de vue logique informatique.

Une solution à ce problème consiste à écrire les dates de façons complètes afin de comparer le caractère '1' avec '0' (48).

Listing 5-9 : Nouvelle comparaison

```
$date1 = "2008/03/07";
$date2 = "2008/12/07";
if ($date1 < $date2) {
  print("Le ".$date1." précéde le ".$date2);
}
else {
  print("Le ".$date1." succède au ".$date2);
}
```

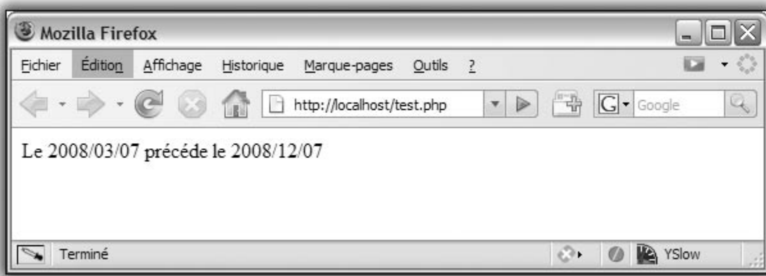


Figure 5.8 : *Résultat désormais valide*

Cette solution n'est cependant pas optimale dans la mesure où nul ne peut garantir que les dates reçues au sein du script seront au bon format.

La véritable solution consiste à comparer non pas des chaînes de caractères mais des timestamps.

Listing 5-10 : Comparaison de timestamps

```
$date1 = "2008/3/7";
$date2 = "2008/12/7";
if (strtotime($date1) < strtotime($date2)) {
  print("Le ".$date1." précéde le ".$date2);
}
else {
  print("Le ".$date1." succède au ".$date2);
}
```

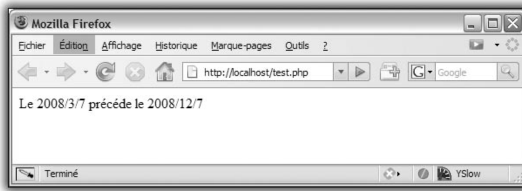


Figure 5.9 : *Résultat désormais valide avec une comparaison de timestamps*

5.2. Formatage d'une date

La fonction `date()` permet d'obtenir une multitude de représentations d'un timestamp. Le premier argument de cette fonction correspond à un format d'affichage et le second à un timestamp.

Le timestamp actuel est utilisé en cas d'absence d'un second argument.

Le format est une simple chaîne de caractères pouvant inclure certaines lettres qui seront automatiquement remplacées par une valeur associée. La lettre `N` par exemple représente le numéro du jour de la semaine (entre 1 et 7).

Listing 5-11 : Affichage du nom du jour actuel

```
$jours = array("lundi", "mardi", "mercredi",
               "jeudi", "vendredi", "samedi",
               "dimanche");
$numero_jour = date("N");
echo "nous sommes un ".$jours[$numero_jour - 1];
```

Associée à `mktime()`, la fonction `date()` permet de savoir quel jour tombait le 1er janvier 2000.

Listing 5-12 : Le 1er janvier 2000

```
$jours = array("lundi", "mardi", "mercredi",
               "jeudi", "vendredi", "samedi",
               "dimanche");
$ts = mktime(0, 0, 0, 1, 1, 2000);
$numero_jour = date("N", $ts);
echo "le 1er janvier 2000 tombait un ".$jours[$numero_jour
✂ - 1];
```

Le format peut contenir plusieurs caractères de substitution. Le format `"Ymd"` permet par exemple d'obtenir le trinôme année, mois, jour sans caractère d'espacement.

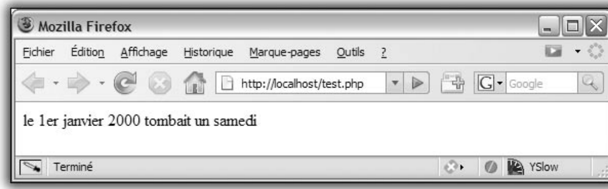


Figure 5.10 : *Il s'agissait d'un samedi*

Listing 5-13 : Plusieurs caractères de substitution au sein d'un même format

```
echo date("Ymd");
// affiche 20080311
```

Les lettres non reconnues (telles que /) au sein du format sont affichées telles quelles.

Listing 5-14 : Le caractère / n'est pas substitué

```
echo date("Y/m/d");
// affiche 2008/03/11
```

Tableau 5.1 : *Caractères de substitution*

Caractère	Description	Exemple/Valeur
a	matinée ou après midi	"am" ou "pm"
A	matin ou après midi (majuscule)	"AM" ou "PM"
B	heure « Internet Swatch »	De 000 à 999
c	date au format ISO 8601	"2004-02-12T15:19:21 +00:00"
d	jour du mois, sur deux chiffres (éventuellement avec un zéro)	"01" à "31"
D	jour de la semaine, en trois lettres (et en anglais)	"Fri" (pour vendredi)
e	identifiant du fuseau horaire	"UTC", "GMT" etc.
F	mois, textuel, version longue, en anglais	"January" (pour janvier)
g	heure, sur 12 heures, sans les zéros initiaux	"1" à "12"
G	heure, sur 24 heures, sans les zéros initiaux	"0" à "23"

Tableau 5.1 : *Caractères de substitution*		
Caractère	**Description**	**Exemple/Valeur**
`h`	heure, au format 12 h	`"01"` à `"12"`
`H`	heure, au format 24 h	`"00"` à `"23"`
`i`	minutes	`"00"` à `"59"`
`I` (i majuscule)	indique si l'heure d'été est activée	`"0"` ou `"1"`
`j`	jour du mois sans les zéros initiaux	`"1"` à `"31"`
`l` (L minuscule)	jour de la semaine en anglais (version longue)	`"Friday"` (pour vendredi)
`L`	indique si l'année est bissextile	`"0"` ou `"1"`
`m`	mois	`"01"` à `"12"`
`M`	mois, en trois lettres et en anglais	`"Apr"` (pour avril)
`n`	mois sans les zéros initiaux	`"1"` à `"12"`
`O`	différence d'heures avec Greenwich (en heures)	`+0200`
`r`	format de date RFC 822	`"Thu, 21 Dec 2000 16:01:07 +0200"`
`s`	secondes	`"00"` à `"59"`
`S`	suffixe ordinal d'un nombre, en anglais, sur deux lettres	`"th"`, `"nd"`
`t`	nombre de jours dans le mois donné	`"28"` à `"31"`
`T`	fuseau horaire de la machine	`"MET"`
`u`	millisecondes	
`U`	secondes depuis l'époque Unix (1er Janvier 1970)	
`w`	jour de la semaine, numérique	`"0"` (dimanche) à `"6"` (samedi)
`W`	numéro de la semaine dans l'année	`"1"` à `"52"`

Caractère	Description	Exemple/Valeur
`Y`	année à quatre chiffres	`"1999"`
`y`	année à deux chiffres	`"99"`
`z`	jour de l'année	`"0"` à `"366"`
`Z`	décalage horaire en secondes (décalage à l'ouest est négatif, à l'est, positif)	`"– 43200"` à `"43200"`

Tableau 5.1 : Caractères de substitution

Echappement de caractères

Les caractères composant le format peuvent être "échappés" afin d'empêcher la substitution.

Listing 5-15 : Affichage sans échappement

```
echo date("date : c");
```

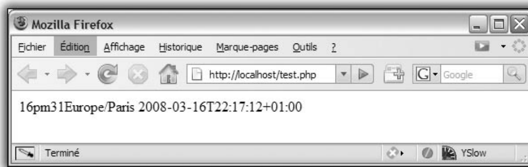


Figure 5.11 : *Toutes les lettres ont été substituées*

Listing 5-16 : Echappement des caractères ne devant pas être substitués

```
echo date("\d\a\t\e : c");
```

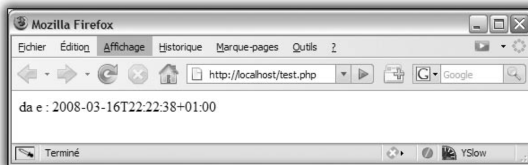


Figure 5.12 : *Les lettres échappées apparaissent (presque) bien*

Toutes les lettres échappées apparaissent bien à l'exception de la lettre `t`. Ce comportement est cohérent avec le chapitre consacré aux chaînes de caractères; la séquence `\t` constitue en effet un caractère spécial au

sein d'une chaîne entre doubles guillemets. La fonction `date()` ne reçoit pas les caractères `\` et `t` mais un caractère de tabulation.

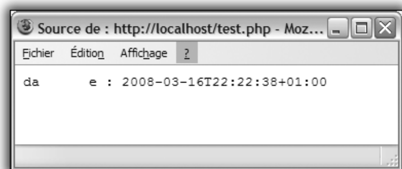


Figure 5.13 : *La séquence \t est bien substituée par une tabulation*

La solution à ce problème consiste simplement à utiliser la prime pour définir le format et éviter ainsi les substitutions de haut niveau réalisées par l'interpréteur.

Listing 5-17 : Utilisation de la prime

```
echo date('\d\a\t\e : c');
```

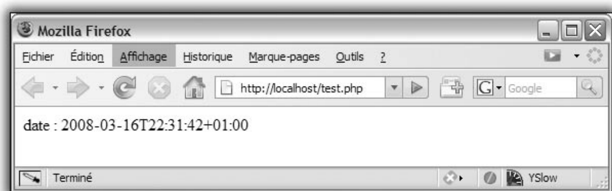


Figure 5.14 : *L'affichage est désormais fidèle au format*

Constantes

Certains formats standard de dates sont prédéfinis dans le cadre de constantes.

Listing 5-18 : Formats standards proposés par PHP

```
echo "DATE_ATOM ".date(DATE_ATOM)."<br/>";
echo "DATE_COOKIE ".date(DATE_COOKIE)."<br/>";
echo "DATE_ISO8601 ".date(DATE_ISO8601)."<br/>";
echo "DATE_RFC822 ".date(DATE_RFC822)."<br/>";
echo "DATE_RFC850 ".date(DATE_RFC850)."<br/>";
echo "DATE_RFC1036 ".date(DATE_RFC1036)."<br/>";
echo "DATE_RFC1123 ".date(DATE_RFC1123)."<br/>";
echo "DATE_RFC2822 ".date(DATE_RFC2822)."<br/>";
echo "DATE_RFC3339 ".date(DATE_RFC3339)."<br/>";
echo "DATE_RSS ".date(DATE_RSS)."<br/>";
echo "DATE_W3C ".date(DATE_W3C)."<br/>";
```

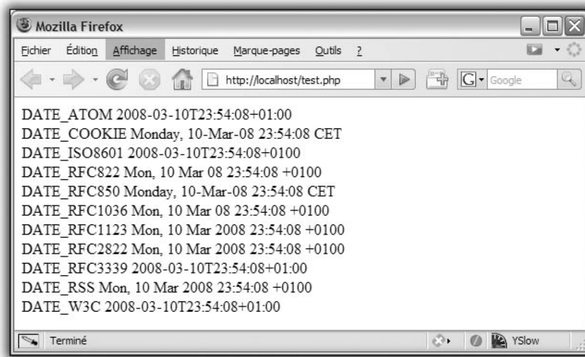


Figure 5.15 : *Différents formats de date*

Ces constantes correspondent très simplement à des chaînes de caractères contenant des caractères substituables dans le cadre de la fonction `date()`.

Listing 5-19 : Contenu des constantes

```
echo "DATE_ATOM : ".DATE_ATOM."<br/>";
echo "DATE_COOKIE : ".DATE_COOKIE."<br/>";
echo "DATE_ISO8601 : ".DATE_ISO8601."<br/>";
echo "DATE_RFC822 : ".DATE_RFC822."<br/>";
echo "DATE_RFC850 : ".DATE_RFC850."<br/>";
echo "DATE_RFC1036 : ".DATE_RFC1036."<br/>";
echo "DATE_RFC1123 : ".DATE_RFC1123."<br/>";
echo "DATE_RFC2822 : ".DATE_RFC2822."<br/>";
echo "DATE_RFC3339 : ".DATE_RFC3339."<br/>";
echo "DATE_RSS : ".DATE_RSS."<br/>";
echo "DATE_W3C : ".DATE_W3C."<br/>";
```

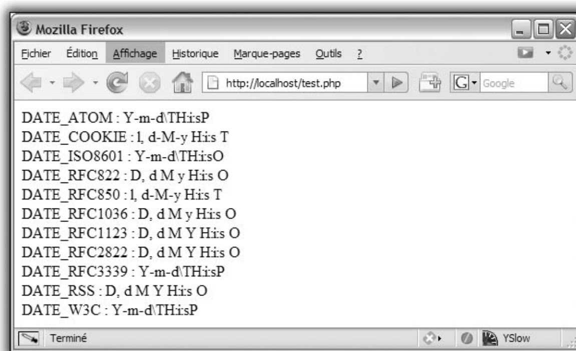


Figure 5.16 : *Détail des formats*

5.3. Contrôle de validité d'une date

Le contrôle de validité d'une date est un problème courant sur le web dans la mesure où les internautes ont en général la possibilité de renseigner directement ce type d'information. Détecter que le 2003/12/32 (32 décembre 2003) est invalide n'est pas un problème très compliqué pour un développeur sachant utilisé la fonction `explode()`. Savoir que le 2007/02/29 (29 février 2007) n'existe pas est en revanche un problème beaucoup plus subtil. La question revient en effet à savoir si l'année 2007 est elle bissextile.

La première solution au problème consiste à utiliser la fonction `strtotime()` pour obtenir un timestamp et ensuite la fonction `date()` pour réaliser l'opération inverse.

Listing 5-20 : Double conversion

```
echo date("Y/m/d", strtotime("2007/02/29"));
```

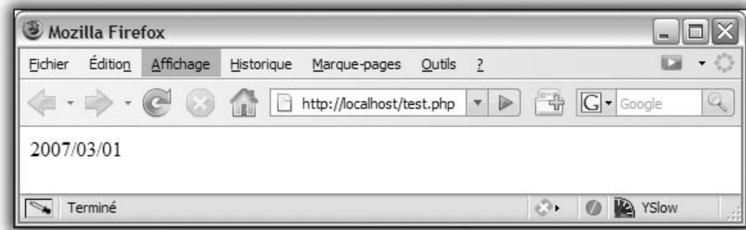


Figure 5.17 : *2007/02/29 est converti en 2007/03/01*

La fonction `strtotime()` a automatiquement converti le 29 février 2007 en 1er mars 2007.

Une simple comparaison permet ainsi de vérifier si la date est valide ou non.

Listing 5-21 : Définition d'une fonction de vérification de date

```
function verif_date($date) {
  $s = date("Y/m/d", strtotime($date));
  return $date == $s ? 1 : 0;
}

echo "2007/02/29 : ".verif_date("2007/02/29")."<br/>\n";
echo "2008/02/29 : ".verif_date("2008/02/29")."<br/>\n";
echo "2008/01/34 : ".verif_date("2008/01/34")."<br/>\n";
```

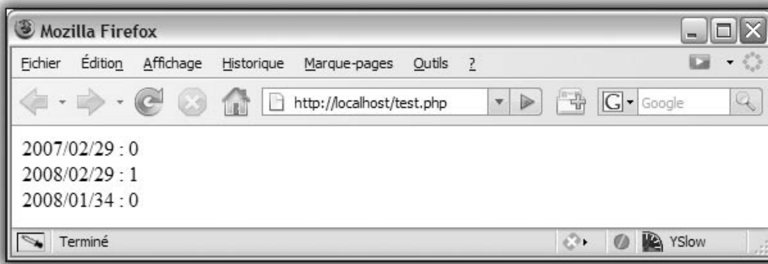


Figure 5.18 : *Les dates invalides sont bien détectées*

Cette méthode a l'inconvénient de reposer sur une comparaison de chaînes de caractères et expose à des erreurs telles que celles vues plus haut.

La véritable solution à ce problème consiste à utiliser la fonction `checkdate()` dont le rôle est précisément de vérifier si une date est valide ou pas. Cette fonction prend trois arguments : le mois, le jour et l'année. La fonction `verif_date()` peut ainsi être réécrite de la manière suivante:

Listing 5-22 : Utilisation de la fonction checkdate()

```
function verif_date($date) {
  list($annee, $mois, $jour) = explode("/", $date);
  return checkdate($mois, $jour, $annee);
}
```

5.4. Check-list

- Le timestamp correspond au nombre de secondes depuis le 1er janvier 1970.
- Une représentation de type timestamp est toujours préférable pour réaliser des opérations au sein d'un code.

Chapitre 6

Les formulaires et transmissions de données

Qu’est-ce qu’un formulaire ? 156
Les différents widgets 158
Passer des paramètres à un script PHP 166
Check-list 181

Nous nous intéresserons, dans le cadre de ce chapitre, aux formulaires HTML. Cette notion est absolument primordiale lors du développement d'un applicatif sur le Web. Ce sont en effet ces formulaires qui vont permettre aux internautes de vous transmettre des informations.

Vous étudierez donc la technique permettant de récupérer et de traiter au sein d'un script PHP des données issues d'un formulaire.

Nous nous arrêterons sur tous les types d'éléments qui peuvent composer de tels formulaires : zones de texte, cases à choix multiples, menus déroulants, etc.

6.1. Qu'est-ce qu'un formulaire ?

Un formulaire peut être envisagé comme un questionnaire permettant aux internautes de transmettre de l'information.

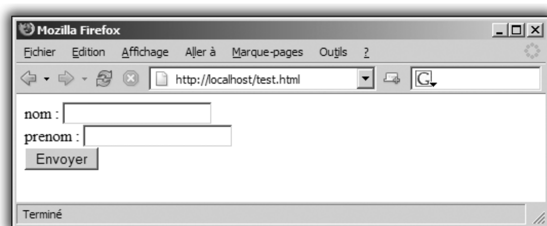


Figure 6.1 : *Exemple de formulaire*

Figure 6.2 : *Formulaire plus complet et stylisé*

Après la validation du formulaire, les données renseignées par l'internaute sont transmises à un script cible. Ce dernier peut aussi bien être écrit en PHP qu'en ASP, en Perl, en Python, etc. Le traitement des données au sein du script pourra alors consister en un envoi de courriel, un enregistrement dans une base de données, la suppression d'un fichier, l'affichage d'une nouvelle page ou toute autre action pouvant être réalisée par le script.

Au niveau structurel, un formulaire est composé de champs. Le premier exemple vous propose un formulaire contenant deux champs : `nom` et `prenom`.

Au niveau HTML, un formulaire se construit ainsi :

- ouverture du formulaire avec la balise `<form>` ;
- définition des champs rattachés à ce formulaire (champs textes, menus, cases à cocher, etc.) ;
- fermeture du formulaire avec la balise `</form>`.

Il est bien sûr possible de proposer plusieurs formulaires par page en multipliant les couples `<form>` `</form>`.



Figure 6.3 : *Deux formulaires*

Le code HTML correspondant à ce dernier formulaire est le suivant :

Listing 6-1 : deux formulaires sur la même page

```
<html>
<body>

<h1>formulaire 1</h1>

<form>
 <label>nom :</label> <input type="text" /><br/>
 <label>prenom :<label> <input type="text" /><br/>
 <input type="submit" />
</form>

<hr/>

<h1>formulaire 2</h1>

<form>
 <label>email :<label> <input type="text" /><br/>
 <input type="submit" />
</form>

</body>
</html>
```

Seuls les éléments directement liés à un formulaire sont transmis par le bouton **Envoyer** du formulaire en question. En validant le deuxième formulaire, seul le champ `email` serait transmis au serveur.

6.2. Les différents widgets

Les champs d'un formulaire peuvent prendre différentes formes. On trouve par exemple des zones de texte, des menus, des cases à cocher, etc. En terme technique, ces champs sont appelés des *widgets*.

Un widget permet de transmettre une information. Chaque widget possède pour cela un nom auquel est associée une valeur.

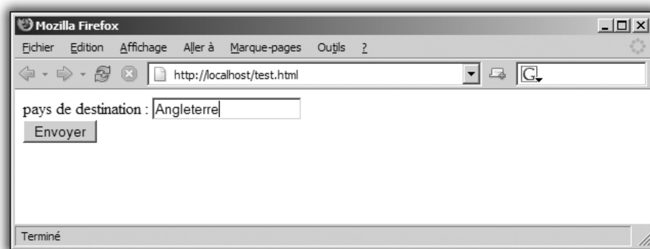


Figure 6.4 : *Le champ texte*

Le code correspondant au champ texte est le suivant :

```
<input type="text" name="pays" />
```

Une fois le formulaire validé par l'internaute, le script cible reçoit une variable `pays` contenant la valeur `Angleterre`.

Arrêtons-nous sur les différents widgets qui peuvent composer un formulaire.

INPUT TEXT



Figure 6.5 : *Widget text*

```
<input type="text" name="nom" />
```

Un widget peut être accompagné d'un certain nombre d'attributs le plus souvent facultatifs. L'attribut `value` par exemple vous permet d'initialiser le champ texte à une valeur donnée.

```
<input type="text" name="nom" value="Dupont" />
```

La dimension

Il est possible de forcer la dimension du champ avec l'attribut `size`.

```
<input type="text" size="50" />
```



Figure 6.6 : *Widget text de taille 50*

La taille maximale

L'attribut `maxlength` permet de limiter le nombre de caractères qui peuvent être renseignés.

```
<input type="text" maxlength="4" />
```

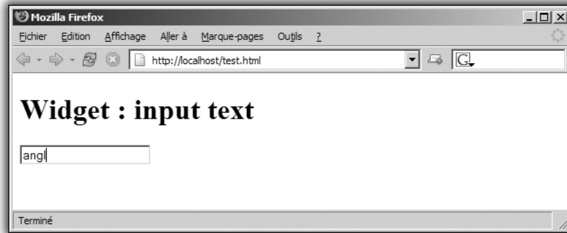


Figure 6.7 : *Avec une taille maximale de 4, il est impossible de taper "angleterre"*

Cet attribut peut être intéressant pour interdire par exemple des identifiants de plus de n caractères.

La lecture seule

L'attribut `readonly` permet d'interdire la modification du contenu du widget.

```
<input type="text" value="donnée protégée" readonly="true" />
```

Cette propriété est disponible pour l'ensemble des widgets présentés ci-après.

Le mot de passe : password

Quand vous souhaitez faire taper un mot de passe à un internaute, il vaut mieux préférer le type `password` au type `text` car les données tapées n'apparaissent pas en clair.

```
<input type="password" name="motdepasse" />
```



Figure 6.8 : *Avec le type "password", les données sont cachées*



XHTML

En HTML standard, les deux lignes suivantes sont équivalentes :

```
<input type="text">
<input type=text>
```

Cependant, la première est compatible avec le format XHTML qui risque de devenir le nouveau standard du Web. Il peut donc être bon, dès maintenant, de prendre de bonnes habitudes. Une autre obligation du XHTML est de fermer les balises « orphelines » :

```
<br /> <img src="pict.gif" /> <input type="text" />
```

Cela ne demande pas beaucoup de travail, ne change rien à l'apparence de votre site et vous assure une compatibilité avec les futurs navigateurs.

TEXTAREA

```
<textarea name="nom"></texarea>
```

Le widget `textarea` correspond à une zone de texte multiligne. Les données transmises par ce champ sont contenues entre les deux balises : `<textarea>` et `</textarea>`.

```
<textarea name="test">ligne 1
ligne 2
</textarea>
```

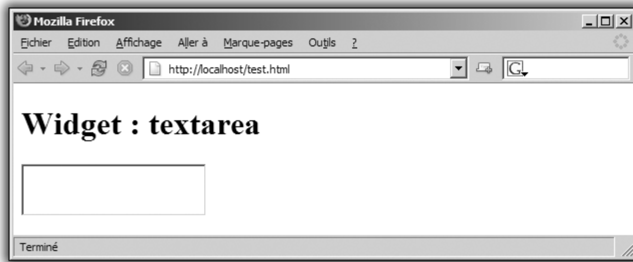


Figure 6.9 : *Textarea*

Les attributs rows et cols

Le premier permet de fixer le nombre de lignes et le second le nombre de colonnes.

Voici un `textarea` de 8 lignes sur 50 colonnes.

```
<textarea name="test" rows="8" cols="50"></textarea>
```

L’attribut wrap

L’attribut `wrap` permet de définir le comportement d’un `textarea` lorsqu’une phrase est trop longue pour tenir en entier dans le widget.

- Avec `wrap="off"`, un ascenseur horizontal apparaît et il est nécessaire de le faire défiler pour voir la fin de la phrase.
- Avec `wrap="soft"`, la phrase tient sur plusieurs lignes.



Figure 6.10 : *Textarea avec et sans wrap*

SELECT

```
<select name="test">
 <option value="1">-- premier choix --</option>
 <option value="2">-- deuxième choix --</option>
</select>
```

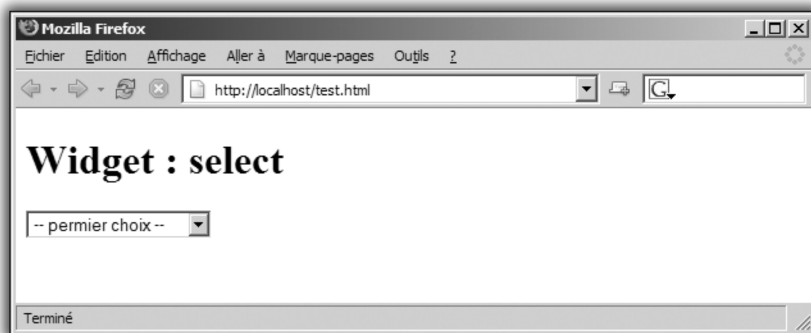


Figure 6.11 : *Le menu déroulant*

Le widget select permet de construire des menus déroulants. Le nom de la donnée est contenu dans la balise <select> et la valeur dans <option>. Seule la valeur sélectionnée sera transmise.

La présélection

Il est possible d'initialiser le menu sur un autre choix que le premier. Il faut alors ajouter le mot-clé selected dans l'option souhaitée :

```
<select name="test">
 <option value="1">-- premier choix --</option>
 <option value="2"
  selected="true">-- deuxième choix --</option>
</select>
```



Figure 6.12 : *Le menu est initialisé sur le deuxième choix*

Le menu à choix multiples

Le mot-clé multiple permet de transformer le menu déroulant en menu à choix multiples. L'attribut associé size permet de définir la hauteur du select.

```
<select name="test[]" multiple="true" size="3">
 <option value="1">-- premier choix --</option>
 <option value="2">-- deuxième choix --</option>
 <option value="3">-- troisième choix --</option>
</select>
```



Figure 6.13 : *Menu à choix multiple*

La encore il est possible d'utiliser `selected` pour présélectionner plusieurs valeurs.

INPUT CHECKBOX

```
<input type="checkbox" name="nom" value="val" />
```

Ce widget permet de réaliser des cases à choix multiples.

```
choix 1 <input type="checkbox" name="choix[]" value="1" /> -
choix 2 <input type="checkbox" name="choix[]" value="2" /> -
choix 3 <input type="checkbox" name="choix[]" value="3" />
```

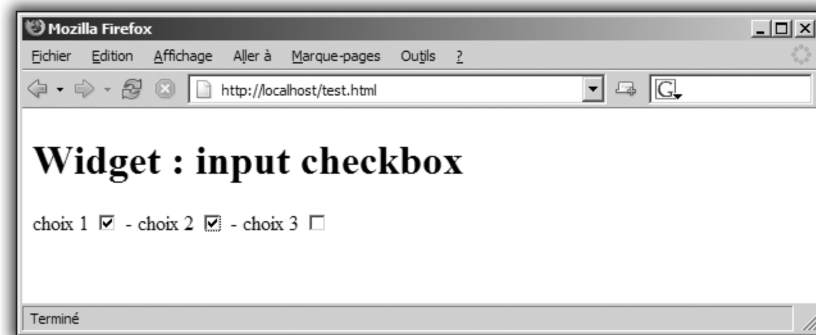


Figure 6.14 : *Les cases à cocher*

Seules les valeurs des cases cochées seront transmises.

Il est possible d'initialiser la case avec le mot-clé `checked`.

```
choix 1 <input type="checkbox" name="choix[]" value="1" /> -
choix 2 <input type="checkbox" name="choix[]"
        value="2" checked="true" /> -
choix 3 <input type="checkbox" name="choix[]" value="3" />
```

INPUT RADIO

```
<input type="checkbox" name="nom" value="val" />
```

À la différence des *checkboxes*, les cases de l'attribut `input radio` sont à choix unique :

```
choix 1 <input type="radio" name="choix[]" value="1" /> -
choix 2 <input type="radio" name="choix[]" value="2" /> -
choix 3 <input type="radio" name="choix[]" value="3" />
```



Figure 6.15 : *Les boutons radios*

Pour en faire des cases à choix unique, il convient de faire attention à bien spécifier le même nom pour tous les attributs `input`. Avec des noms différents, il devient possible de toutes les sélectionner.

```
choix 1 <input type="radio" name="choix1" value="1" /> -
choix 2 <input type="radio" name="choix2" value="2" /> -
choix 3 <input type="radio" name="choix3" value="3" />
```



Figure 6.16 : *Des attributs input avec des noms différents ne s'excluent pas*

Le mot-clé `checked` est aussi valide avec ce widget.

INPUT BUTTON

```
<input type="button" name="nom" value="val" />
```

Il s'agit du widget "bouton". L'intitulé du bouton est contenu dans `value` :

```
<input type="button" name="test" value="ceci est un bouton" />
```

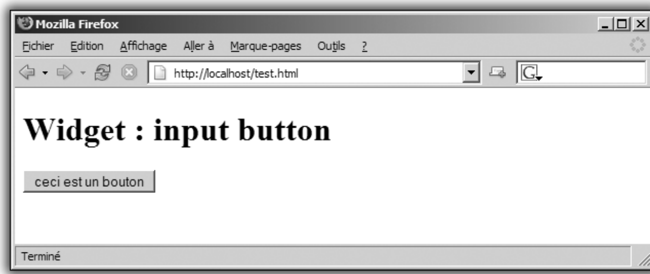


Figure 6.17 : *Le bouton*

Il existe deux « boutons » spéciaux.

- `submit` : c'est en cliquant sur un bouton de ce type que vous déclenchez l'envoi du formulaire.

  ```
  <input type="submit" value="envoyer le formulaire" />
  ```

- `reset` : ce bouton permet d'effacer toutes les informations contenues dans le formulaire.

  ```
  <input type="reset" value="effacer les informations" />
  ```



Une image comme bouton d'envoi

Il est possible de remplacer un bouton `submit` par une image. La syntaxe est alors :

`<input type="image" src="image.gif" />`.

INPUT HIDDEN

```
<input type="hidden" name="nom" value="val" />
```

Il s'agit d'un widget invisible. L'intérêt, me demanderez-vous ? Stocker une information que vous voulez transmettre avec le formulaire, mais qui n'a pas lieu d'être présentée à l'internaute.

6.3. Passer des paramètres à un script PHP

Maintenant que vous êtes en mesure de construire vos propres formulaires, découvrez les différents moyens de transmettre ces informations à un script PHP qui pourra alors les traiter.

La variable $_GET

La balise <form> dispose également d'un certain nombre d'attributs optionnels. Le premier que vous allez étudier est `action`. Il permet de préciser quel script sera appelé une fois le bouton `submit` du formulaire validé.

Si vous souhaitez transmettre les informations du formulaire au script *script.php*, vous devez écrire :

```
<form action="script.php">
```



`form` sans action

Si vous ne précisez pas la propriété action, le formulaire s'envoie les informations à lui-même. Vous verrez plus loin qu'il peut être intéressant de regrouper à la fois le formulaire et le traitement du formulaire au sein d'un même fichier.

Un script situé sur un autre site peut également être appelé via cet attribut `action`. L'exemple suivant appelle le moteur de recherche Google en lui transmettant le mot-clé correspondant à votre recherche :

Listing 6-2 : Appel direct du moteur de recherche Google

```
<form action="http://www.google.fr/search">
 <input type="text" name="q" />
 <input type="submit" value="nouvelle recherche" />
</form>
```



Figure 6.18 : *Formulaire permettant d'appeler le moteur de recherche Google*

En validant le formulaire avec le mot-clé `"PHP"`, vous basculez sur le site de Google et vous obtenez le résultat de la recherche :

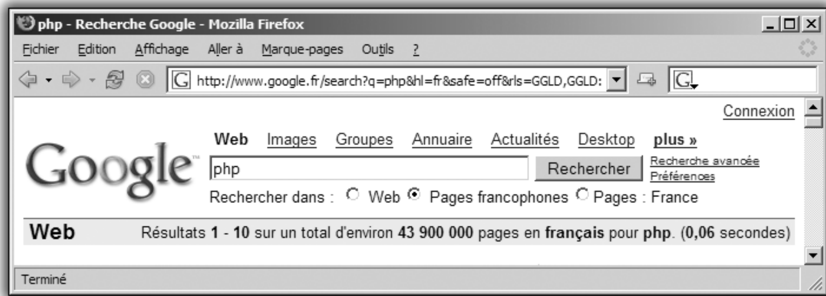


Figure 6.19 : *Résultat de la recherche*



Utilisation de ressources

Le Web est de plus en plus réglementé. Les grandes heures d'Internet libre sont derrière nous. Le fait d'utiliser des ressources d'un autre serveur, comme vous venez de le faire, n'est pas toujours très apprécié.

Créez maintenant deux fichiers en y plaçant un formulaire (*form.html*) et un script PHP (*script.php*) et faites pointer le formulaire sur le script :

Listing 6-3 : form.html

```
<form action="script.php">
 <input type="text" name="x" />
 <input type="submit" value="envoyer" />
</form>
```

Listing 6-4 : script.php

```
<?php
print("valeur de x : ");
?>
```

En validant votre formulaire, le script affiche le message `"bonjour monde"`. C'est certes sympathique, mais vous ne prenez pas en compte l'information qui a été transmise.

PHP rend la récupération des paramètres transmis extrêmement simple. Le principe consiste à passer par la variable prédéfinie `$_GET` qui contient un tableau associatif dont chaque mot-clé correspond au nom

(attribut `name`) du widget. Dans le cas présent, `$_GET['x']` contient la valeur de votre champ texte.

Listing 6-5 : votre script affiche maintenant la valeur qui lui a été transmise

```
print("valeur de x : "+$_GET['x']);
```

La variable `$_GET` est d'un type particulier qui lui permet d'être directement accessible au sein de fonctions sans passer par `global`. Le terme utilisé pour la qualifier est « super-global » .

Listing 6-6 : script.php

```
<?php

function affiche_param()
{
  print("valeur de x : "+$_GET['x']);
}

affiche_param();

?>
```

Le principe est le même, qu'il y ait un ou plusieurs champs et quel que soit le type du champ. Réalisez un formulaire plus complexe permettant de travailler avec différents widgets :

Listing 6-7 : un formulaire plus complet

```
<html>
<body>

<form action="script.php">

<label>Titre du film</label>
<input type="text" name="titre" /><br/>

<label>Genre</label>
<select name="genre">
 <option value="policier">POLICIER</option>
 <option value="sf">SCIENCE FICTION</option>
</select><br/>

<label>Description</label>
<textarea name="description"></textarea><br/>

<label>Film en couleur</label>
<input type="radio" name="couleur" value="1" /> oui -
<input type="radio" name="couleur" value="non" /> non <br/>
```

```
<br/>
<input type="submit" value="envoyer" />

</form>

</body>
</html>
```

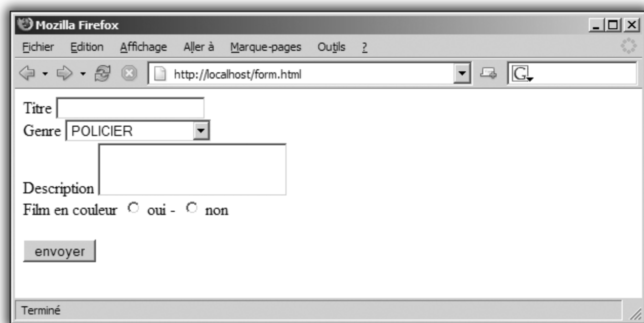


Figure 6.20 : *Formulaire complet*

Modifiez le script *script.php* afin d'afficher toutes les données transmises par votre formulaire :

Listing 6-8 : script.php

```
<?php

print("<b>Titre</b> : ".$_GET['titre']."<br/>");
print("<b>Genre</b> : ".$_GET['genre']."<br/>");
print("<b>Description</b> : ".$_GET['description']."<br/>");
print("<b>Couleur ?</b> : ".$_GET['couleur']."<br/>");

?>
```

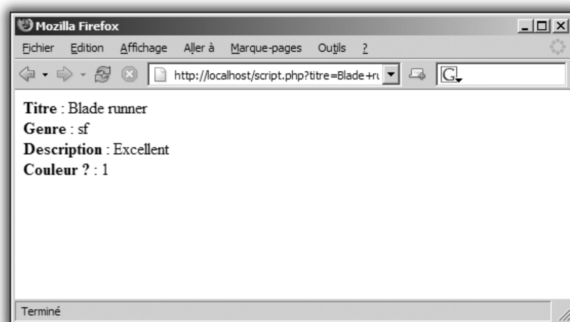


Figure 6.21 : *Affichage des données*



Ouverture dans une autre fenêtre

Il est possible en HTML d'ouvrir un lien dans une autre fenêtre avec l'attribut `target` :

```
<a href="http://www.google.fr"
target="_blank">google</a>.
```

Cela fonctionne de la même manière avec les formulaires :

```
<form action="script.php" target="_blank">.
```

Votre formulaire n'est composé pour l'instant que de widgets à choix unique. Vous avez cependant pu lire plus haut qu'il existe des widgets permettant de sélectionner plusieurs valeurs : les attributs multilignes `checkbox` et `select`. L'association une variable/plusieurs valeurs vous conduit naturellement aux tableaux ! La technique permettant d'indiquer à PHP d'initialiser un tableau plutôt qu'une variable simple consiste à mettre des crochets au sein de l'attribut `name` : vous écrirez `name="val[]"` plutôt que `name="val"`.

Modifiez *form.html* afin d'y intégrer des widgets à choix multiples.

Listing 6-9 : formulaire avec des champs à choix multiples

```
<html>
<body>

<form action="script.php">

<label>Titre du film</label>
<input type="text" name="titre" /><br/>

<label>Genre</label>
<select name="genre[]"multiple="yes" size="3">
 <option value="policier">POLICIER</option>
 <option value="sf">SCIENCE FICTION</option>
 <option value="culte">CULTE</option>
</select><br/>


<label>Description</label>
<textarea name="description"></textarea><br/>

<label>Film en couleur</label>
<input type="radio" name="couleur" value="1" /> oui -
<input type="radio" name="couleur" value="non" /> non <br/>
```

```
<label>Pays</label>
<select name="pays">
 <option value="fr">FRANCE</option>
 <option value="us">USA</option>
 <option value="gb">ANGLETERRE</option>
</select><br/>

<label>Sous titre</label>
<input type="checkbox" name="soustitre[]" value="fr" />
français -
<input type="checkbox" name="soustitre[]" value="gb" />
anglais -
<input type="checkbox" name="soustitre[]" value="es" />
espagnol <br/>
<br/>
<input type="submit" value="envoyer" />

</form>

</body>
</html>
```

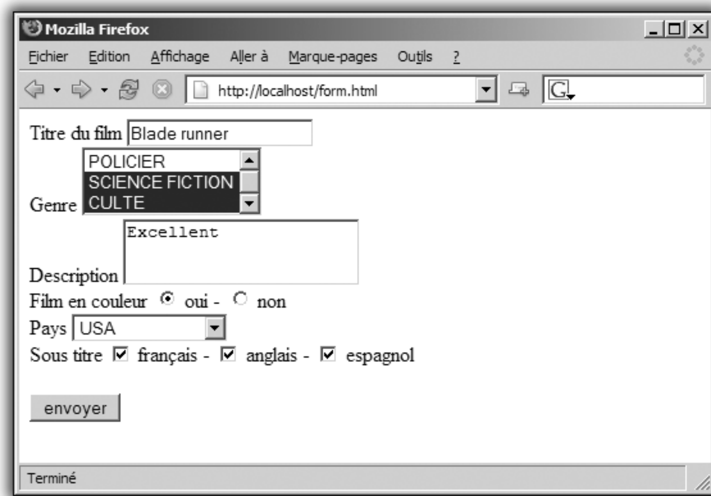


Figure 6.22 : *Formulaire avec des champs à choix multiples*

Affichez le contenu de la variable `$_GET` afin de comprendre comment PHP organise la réception des données. Utilisez pour cela la fonction `print_r()` qui, comme vous l'avez vu précédemment, permet d'afficher le contenu d'un tableau.

Listing 6-10 : affichage de $_GET

```
<pre>
<?php
print_r($_GET);
?>
</pre>
```

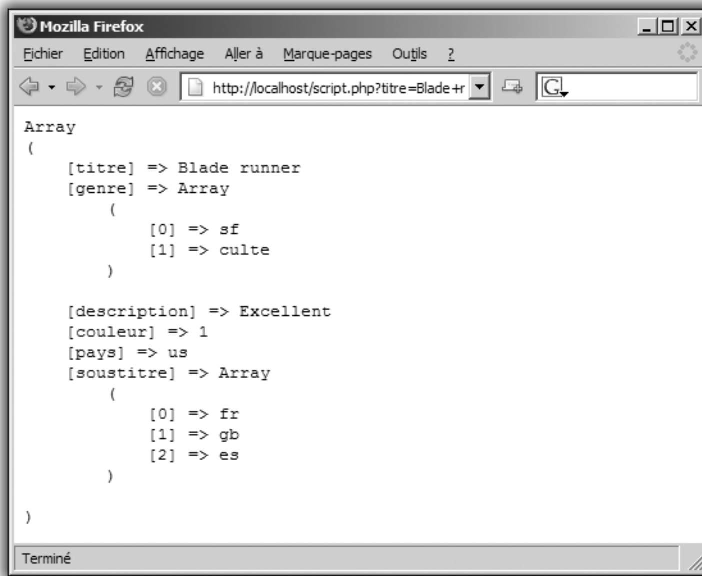


Figure 6.23 : *Contenu de la variable $_GET*



Balise `<pre>`

Le texte placé entre les balises `<pre>` et `</pre>` est affiché en tant que texte brut. Cela permet, dans le cadre de la fonction `print_r()`, de conserver les sauts de ligne et l'indentation.

Vous obtenez la confirmation que les éléments `$_GET['genre']` et `$_GET['soustitre']` contiennent bien des tableaux. Il convient donc de trouver un moyen de les afficher au sein de votre script *script.php*. La fonction `join()` qui convertit un tableau en chaîne de caractères pourrait tout à fait convenir. Rappelons en effet son utilisation à l'aide d'un exemple :

```
$ar = array('a','b','c');
$str = join(',',$ar);
print($str);
```

Ce morceau de code affichera : `a,b,c`.

Votre script devient donc :

Listing 6-11 : affichage des champs multivaleurs

```
<?php

print("<b>Titre</b> : ".$_GET['titre']."<br/>");
$str_genre = join(',',$_GET['genre']);
print("<b>Genre</b> : ".$str_genre."<br/>");
print("<b>Description</b> : ".$_GET['description']."<br/>");
print("<b>Couleur</b> : ".$_GET['couleur']."<br/>");
print("<b>Pays</b> : ".$_GET['pays']."<br/>");
$str_soustitre = join(',',$_GET['soustitre']);
print("<b>Sous titres</b> : ".$str_soustitre."<br/>");

?>
```

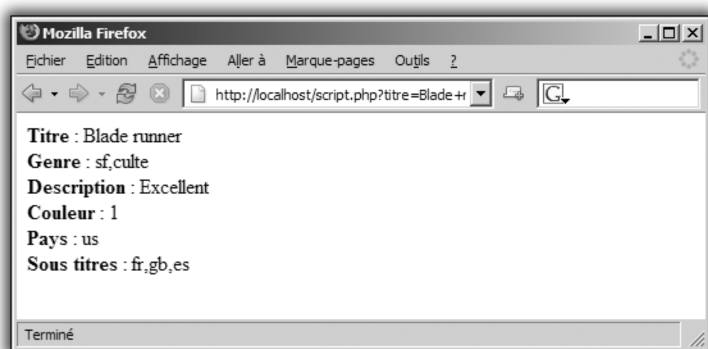


Figure 6.24 : *Affichage des valeurs*

Query String

Arrêtons-nous un instant sur l'adresse (URL) affichée en haut du navigateur dans la barre de navigation :

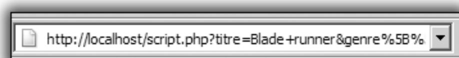


Figure 6.25 : *Affichage d'une URL*

Vous constatez que le formulaire appelle bien *script.php* et qu'il lui transmet en plus une longue chaîne de caractères assez étrange : `http://localhost/script.php?titre=Blade+runner&genre 5B%5D=sf&genre%5B%5D=culte&description=Excellent &couleur=1&pays=us&soustitre%5B%5D=fr&soustitre %5B%5D=gb&soustitre5B5D=es`.

La caractéristique principale de la méthode `GET` est précisément de transmettre les paramètres du formulaire au sein même de l'URL. Cette chaîne de paramètres (dont le nom technique est *Query String*) est séparée du nom du script par le caractère `?`. Le codage permettant de la composer se révèle assez facile :

```
nom1=valeur1&nom2=valeur2&nom3=valeur3…nomN=valeurN
```

où `nom` correspond à l'attribut `name` du widget et `valeur` correspond à la valeur renseignée, sélectionnée ou cochée par l'internaute.

L'URL suivante confirme la chose car vous y trouvez bien `description=Excellent`. En revanche, le caractère + dans `titre=Blade+runner` montre que les valeurs subissent elles-mêmes un autre codage. Il s'agit de l'URL d'encodage qui transforme par exemple les espaces en signes +, les crochets (`[`) en `%5B`, les signes = en `%3D`, les `é` en `%E9`, etc. La fonction PHP `urlencode()` se charge de cette opération fastidieuse en retournant une chaîne `urlencode` à partir d'une chaîne « en clair ».

```
$str = urlencode("a = é & 123");
// $str contient a+%3D+%E9+%26+123
```



URL encodage de l'attribut
À la fois les noms et les valeurs doivent être encodés dans l'URL !

Vous comprenez qu'il est maintenant possible de transmettre des données à votre script sans passer par le formulaire. Imaginez que vous vouliez transmettre le film suivant :

Tableau 6.1 : *Film Manhattan*

Caractéristiques	**Valeur**
Titre	`Manhattan`

Tableau 6.1 : *Film Manhattan*

Caractéristiques	Valeur
Genre	culte
Description	magnifique
Couleur	0
Pays	us
Sous titres	fr, gb

Écrivez maintenant le script *url.php* chargé de construire un lien permettant de transmettre à *script.php* les données liées au film *Manhattan*.

Listing 6-12 : transmission de données au sein de l'URL

```
<?php

$url  = 'http://localhost/script.php?';
$url .= urlencode("titre").'='.urlencode("Manhattan").'&';
$url .= urlencode("genre[]").'='.urlencode("culte").'&';
$url .= urlencode("description").'='.urlencode("magnifique").'&';
$url .= urlencode("couleur").'='.urlencode("0").'&';
$url .= urlencode("pays").'='.urlencode("us").'&';
$url .= urlencode("soustitre[]").'='.urlencode("fr").'&';
$url .= urlencode("soustitre[]").'='.urlencode("gb");

print("<a href='".$url."'>Film Manhattan</a>");

?>
```

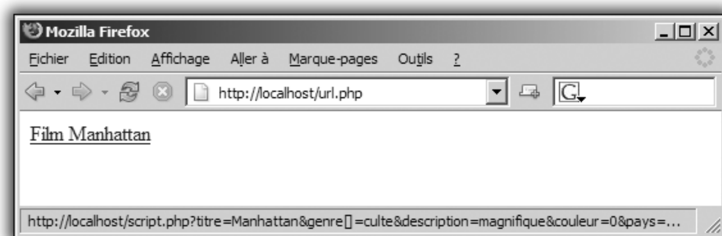


Figure 6.26 : *La barre d'état montre que l'URL a bien été construite*

Cliquez maintenant sur le lien, et vous obtenez le film attendu :

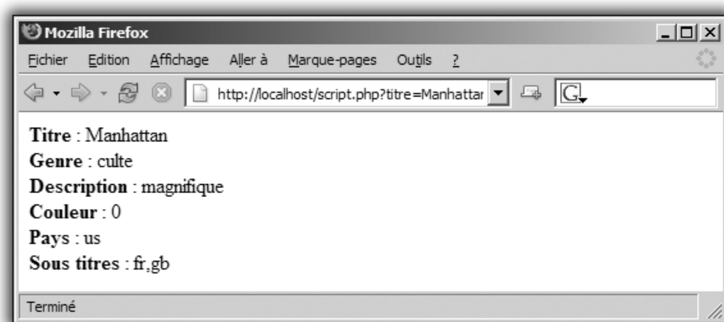


Figure 6.27 : Affichage des résultats



Transmission simple

Si les noms et les valeurs à transmettre ne contiennent que des caractères alphanumériques, l'étape d'encodage URL devient superficielle. Pour x = 2 et y = "toto", le lien suivant peut directement être écrit : transmission simple.

Ce type de transmission par URL se révèle très pratique en programmation web car elle évite de mettre systématiquement en œuvre un formulaire pour transmettre une donnée.

La méthode POST

La méthode GET entraîne donc l'affichage en clair dans la barre de navigation des données renseignées par l'internaute.

Vous comprenez donc la limitation d'une telle méthode dans le cas de la transmission d'un mot de passe.

Listing 6-13 : formulaire d'identification

```
<form action="auth.php">

 <label>identifiant</label>
 <input type="text" name="id" /><br/>

 <label>mot de passe</label>
 <input type="password" name="pass" /><br/>
```

```
 <br/><input type="submit" value="identification">
</form>
```

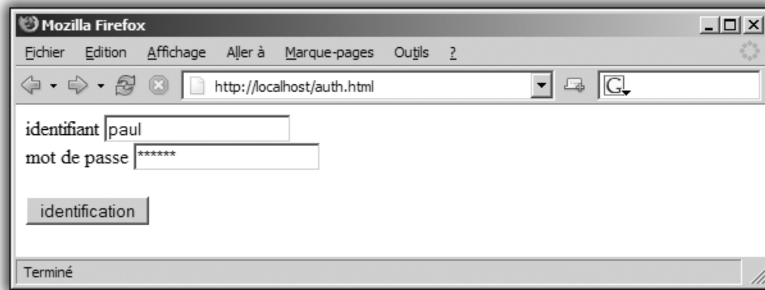


Figure 6.28 : *Formulaire d'identification disposant d'un widget password*

Listing 6-14 : auth.php

```
<?php

print("<b>Identifiant</b> : ".$_GET['id']."<br/>");
print("<b>Mot de passe</b> : ".$_GET['pass']."<br/>");

?>
```

En validant le bouton d'identification, vous basculez sur le script *auth.php* et vous aurez la surprise de voir dans la barre de navigation votre mot de passe en clair !

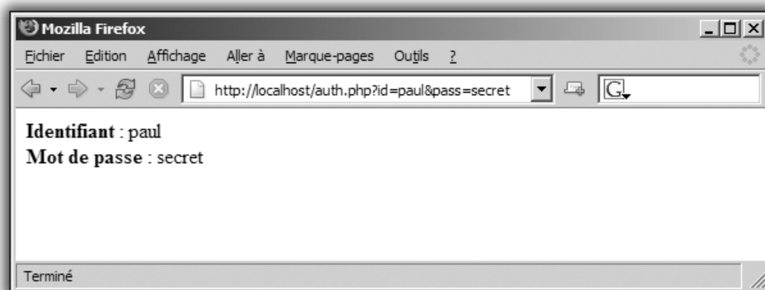


Figure 6.29 : *Il faut espérer qu'un collègue mal intentionné ne se trouve pas derrière !*

Heureusement, les formulaires peuvent également être envoyés avec la méthode `POST` !

Listing 6-15 : Le formulaire utilise désormais la méthode post

```
<form action="auth.php" method="post">

 <label>identifiant</label>
 <input type="text" name="id" /><br/>

 <label>mot de passe</label>
 <input type="password" name="pass" /><br/>

 <br/><input type="submit" value="identification">

</form>
```

En termes fonctionnels, cela ne change presque rien pour vous : la variable `$_POST` est utilisée plutôt que la variable `$_GET`.

```
<?php

print("<b>Identifiant</b> : ".$_POST['id']."<br/>");
print("<b>Mot de passe</b> : ".$_POST['pass']."<br/>");

?>
```

En revanche, les paramètres transmis n'apparaissent plus dans la barre de navigation.

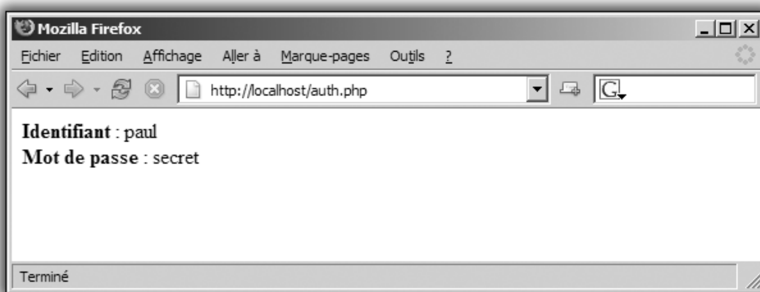


Figure 6.30 : *Le mot de passe n'apparaît plus dans la barre de navigation*



REMARQUE

Comment choisir entre POST et GET

La méthode `POST` peut être préférée à la méthode `GET` lorsque vous devez transmettre une quantité importante de données (plusieurs mégaoctets). Inversement, il est conseillé de l'éviter si l'internaute est susceptible de revenir en arrière en utilisant l'icône de retour (*back*). Vous risquez en effet dans ce cas d'être confronté à des problématiques de « cache » et de *timeout*.



Plus généralement, la méthode POST est conseillée lorsque le script de destination modifie une donnée et que le rafraichissement de la page et la mise en cache de la requête ne sont pas souhaités.

Il est donc recommandé de mettre à jour un enregistrement d'une base de données en POST et de réaliser une recherche avec la méthode GET.

Vous constatez qu'un changement de méthode (POST, GET) au niveau de la balise <form> entraîne une modification du script. La variable $_REQUEST permet d'éviter cela dans la mesure où elle contient à la fois les variables transmises en POST, les variables transmises en GET ainsi que les cookies (que vous étudierez plus loin). Vous ferez donc le choix dans le reste de l'ouvrage de passer par cette variable.

Le mode register_globals on

Pour les versions plus anciennes de PHP, la récupération des variables était encore plus simple. Des variables portant le nom des widgets étaient automatiquement initialisées.

Vous auriez écrit *script.php* de la manière suivante :

Listing 6-16 : ancienne méthode pour récupérer les paramètres

```
<?php

print("<b>Titre</b> : ".$titre."<br/>");
$str_genre = join(',',$genre[]);
print("<b>Genre</b> : ".$str_genre."<br/>");
print("<b>Description</b> : ".$description."<br/>");
print("<b>Couleur</b> : ".$couleur."<br/>");
print("<b>Pays</b> : ".$pays."<br/>");
$str_soustitre = join(',',$soustitre[]);
print("<b>Sous titres</b> : ".$str_soustitre."<br/>");

?>
```

Il est encore possible d'utiliser cette façon d'opérer en initialisant à on la directive de configuration register_globals. Nous vous le déconseillons cependant vivement afin de limiter au maximum les failles de sécurité que cela entraîne bien souvent.



La fonction `ini_get()`

La fonction `ini_get()` vous permet de récupérer la valeur d'une directive de configuration PHP et ainsi de savoir si vous êtes en `register_globals` `on` ou `off` :

```
<?php

if (ini_get("register_globals")==1) {
 print("Vous êtes en register_globals=on");
}
else {
 print("Vous êtes en register_globals=off");
}

?>
```

6.4. Check-list

- Les formulaires permettent de récupérer de l'information auprès des internautes et de les transmettre à un script PHP cible.
- Seuls les champs situés entre `<form>` et `</form>` sont transmis au moment de la validation du formulaire.
- L'attribut `action` du `FORM` permet de spécifier l'emplacement du script cible.
- Deux modes de transmission sont possibles pour les formulaires : `GET` et `POST`. Le mode est précisé à l'aide de l'attribut `method`.
- Un formulaire est composé d'un certain nombre de champs. Ces derniers peuvent prendre différentes formes : champ texte, champ texte multiligne, menu déroulant, cases à cocher, etc.
- Des paramètres peuvent également être transmis directement dans l'URL. On parle alors de *Query String*.
- Au sein du script, les paramètres peuvent être récupérés avec la variable `$_REQUEST[]`.

Chapitre 7

En tête HTTP et authentification

Requêtes et réponses 184
Fonction header() 188
Page d’erreur 190
Authentification 192
En bref 196

Le web repose sur le protocole HTTP. Ce dernier permet à un navigateur comme Firefox de communiquer avec un serveur tel qu'Apache. Cet échange est bien évidemment normé et codifié.

Ce chapitre vise à mieux appréhender ces échanges et à démontrer qu'une bonne connaissance de ce protocole permet de tirer partie de fonctionnalités avancées du couple navigateur/serveur.

7.1. Requêtes et réponses

La demande faite par un navigateur à un serveur web est appelée une requête (`HTTP Request`). Une requête typique consiste à demander le contenu d'un page ou d'une image. Suite à cette requête, le serveur retourne une réponse (`HTTP Response`). Requêtes et réponses sont composées d'une multitude d'instructions formant une en tête.

Extension LiveHTTPHeaders

L'extension de Firefox *LiveHTTPHeaders* permet de visualiser précisément les échanges HTTP entre un client et un serveur. Elle peut être téléchargée gratuitement sur le site **http://livehttpheaders.mozdev.org**. en utilisant les liens *Installation* puis *Install version 0.X of LiveHTTPHeaders now*. Firefox vous propose alors d'ajouter ce site à sa liste de sites auxquels il peut faire confiance.

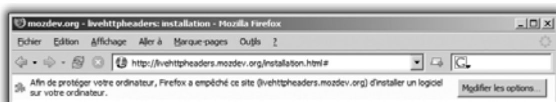


Figure 7.1 : *Utilisez le bouton "Modifier les options"*

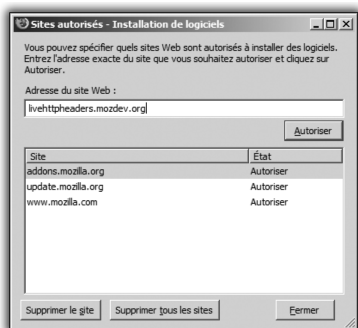


Figure 7.2 : *Utilisez le bouton "Autoriser"*

Une fois ce domaine ajouté cliquez à nouveau sur le lien *Install version 0.XX of LiveHTTPHeaders now* pour installer véritablement l'extension.

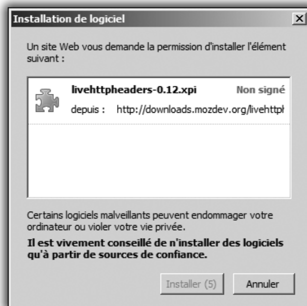


Figure 7.3 :
Utilisez le bouton "Installer" une fois le décompte terminé

Le navigateur doit ensuite être fermé puis réouvert pour prendre en compte cette nouvelle extension.

La première utilisation de cet outil va permettre d'espionner l'échange suivant : le navigateur appelle l'URL **http://localhost/test.php**, et le serveur retourne le résultat de l'interprétation du script suivant :

```
<?php
print("ok");
?>
```

L'extension peut être lancée en passant par le menu **Outils / En têtes HTTP en direct**.

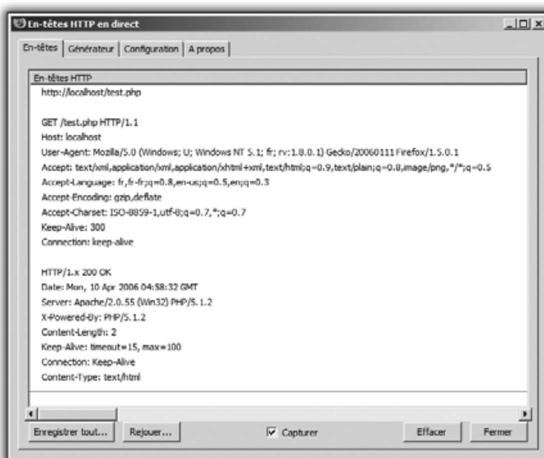


Figure 7.4 : *Visualisation d'un échange requête / réponse*

La requête

Trois catégories de données sont incluses dans l'en tête de la requête :

- des informations sur la page cible et la version du protocole

```
GET /test.php HTTP/1.1
Host: localhost
```

- des précisions sur le navigateur

```
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1;
✂ fr; rv:1.8.0.1) Gecko/20060111 Firefox/1.5.0.1
Accept: text/xml,application/xml,application/xhtml+xml,
✂ text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
```

- le type de la connexion

```
Keep-Alive: 300
Connection: keep-alive
```

Ces informations peuvent être récupérées au sein du script à l'aide de la fonction `getallheaders()` qui retourne un tableau contenant l'ensemble des informations présentes dans l'en-tête.

Ces informations permettent au script d'ajuster son comportement en fonction :

- du type du navigateur (`User–Agent`),
- des formats des fichiers acceptés (`Accept`),
- de la langue (`Accept–Language`).

L'exemple suivant affiche un message de reconnaissance de votre navigateur.

Listing 7-1 : Détection du navigateur

```
$header = getallheaders();

if (strpos($header["User-Agent"], "Firefox") > 0) {
  print("Vous utilisez Firefox");
}
elseif (strpos($header["User-Agent"],"MSIE") > 0) {
  print("Vous utilisez Internet Explorer");
}
```

```
else {
  print("Navigateur non reconnu");
}

print("<br/><br/><i>".$header["User-Agent"]."</i>");
```

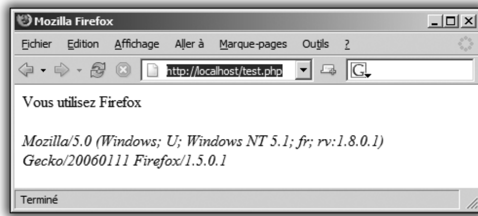


Figure 7.5 : *Détection de Firefox*

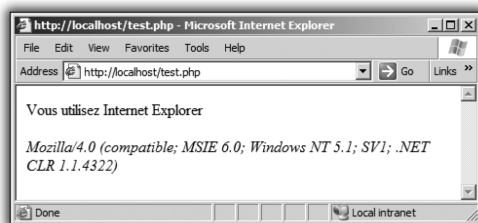


Figure 7.6 : *Détection d'Internet Explorer*

La réponse

L'en-tête de la réponse donne des informations sur :

- le code réponse,

```
HTTP/1.x 200 OK
```

- le serveur distant,

```
Date: Mon, 10 Apr 2006 04:58:32 GMT
Server: Apache/2.0.55 (Win32) PHP/5.1.2
X-Powered-By: PHP/5.1.2
```

- le contenu,

```
Content-Length: 2
Content-Type: text/html
```

- la connexion,

```
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
```

Le contenu à proprement parler fait suite à l'en-tête dans le message de réponse retourné par le serveur web.

La fonction `apache_response_headers()` retourne un tableau associatif contenant tous les éléments de l'en-tête de réponse.

7.2. Fonction header()

PHP permet de modifier l'en tête de la réponse à l'aide de la fonction `header()`. L'emplacement de cette fonction au sein d'un script est extrêmement important : la fonction `header()` doit absolument être placée avant le premier affichage de la page.

Les deux scripts suivants affichent le message *« Bonjour »* avant de faire appel à `header()`. Ils provoquent par conséquent l'affichage d'une erreur.

Listing 7-2 : Affichage provoquant une erreur

```
Bonjour
<?php
header("Text: coucou");
?>
```

Listing 7-3 : Autre syntaxe provoquant une erreur

```
<?php
print("Bonjour");
header("Text: coucou");
?>
```

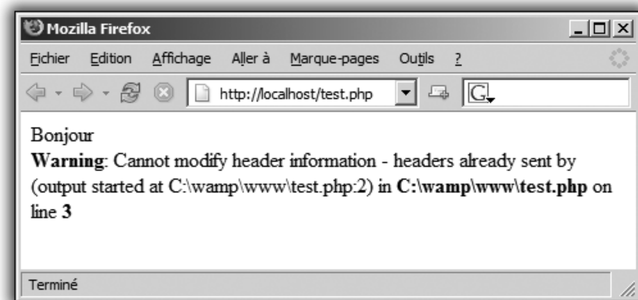


***Figure 7.7 :** Erreur générée suite à un mauvais placement de la fonction header()*



Prudence

Un simple retour à la ligne ou un espace situé avant la balise `<?php` conduiraient également à une erreur ! La fonction `header()` impose une grande précision.

En déplaçant l'affichage derrière la fonction `header()`, l'erreur disparaît et vous constatez que l'en-tête a bien été mise à jour.

Listing 7-4 : Exemple valide

```
<?php
header("Text: coucou");
print("Bonjour");
?>
```

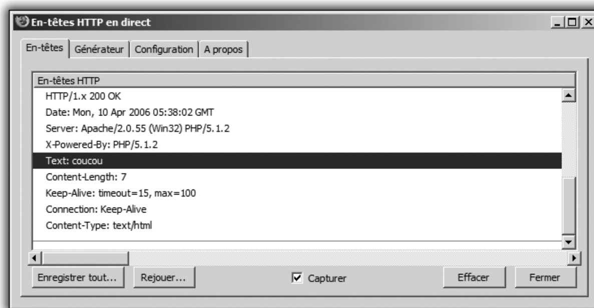


Figure 7.8 : *L'en tête a bien été modifiée*

Il est également possible de modifier le contenu des instructions en les écrasant (voir Figure 7.9) :

Listing 7-5 : Modification de l'instruction X-Powered-By

```
<?php
print("Bonjour");
header("X-Powered-By: PHP/8 !!!");
?>
```

Plus utile, la fonction `header()` permet également de donner des instructions telle qu'une demande de redirection avec l'instruction `Location`.

Listing 7-6 : Demande de redirection vers une autre page

```
<?php
header("Location: http://www.google.com");
?>
```

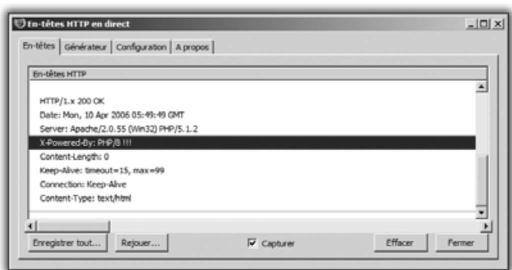


Figure 7.9 : *Modification de l'instruction X-Powered-By*

7.3. Page d'erreur

Le code retour `200` présent dans l'en tête de réponse (`HTTP/1.x 200 OK`) signifie que le navigateur n'a rencontré aucun problème pour traiter la requête.

En réalisant une requête sur une page n'existant pas sur le serveur, le code d'erreur passe à `404`.



Figure 7.10 : *Page d'erreur affichée par le client*

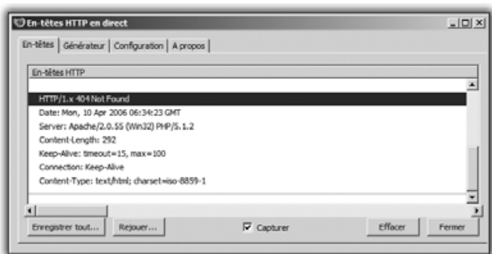


Figure 7.11 : *En tête d'erreur*

Le protocole HTTP prévoit un certain nombre de codes d'erreur tous commençant par 4xx.

Tableau 7.1 : *Codes d'erreur client*

Code d'erreur	Signification
400	Erreur de syntaxe dans l'adresse du document
401	Pas d'autorisation d'accès au document
402	Accès au document soumis au paiement
403	Pas d'autorisation d'accès au serveur
404	La page demandée n'existe pas
405	Méthode de requête du formulaire non autorisée
406	Requête non acceptée par le serveur
407	Autorisation du proxy nécessaire
408	Temps d'accès à la page demandée expiré
409	L'utilisateur doit soumettre à nouveau avec plus d'infos
410	Cette ressource n'est plus disponible
411	Le serveur a refusé la requête car elle n'a pas de longueur
412	La pré condition donnée dans la requête a échoué
413	L'entité de la requête était trop grande
414	L'URI de la requête était trop longue
415	Type de média non géré

Vous pouvez vous-même déclencher les erreurs que vous souhaitez en retournant la valeur du code d'erreur avec la fonction header().

L'exemple suivant indique que la page n'existe pas uniquement aux visiteurs utilisant Firefox :

Listing 7-7 : Détection du navigateur en PHP

```
$header = getallheaders();

if (strpos($header["User-Agent"], "Firefox") > 0) {
  header("HTTP/1.0 404 Not Found");
  print("La page n'existe pas");
}
else print("Hello IE");
```

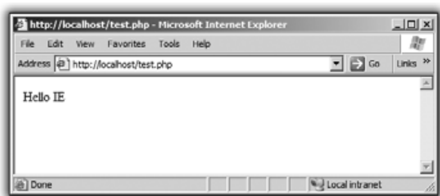


Figure 7.12 : La page sans erreur avec Internet Explorer

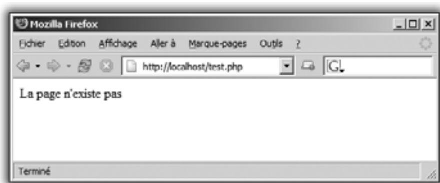


Figure 7.13 : L'utilisation de Firefox en revanche déclenche l'apparition d'un message d'erreur

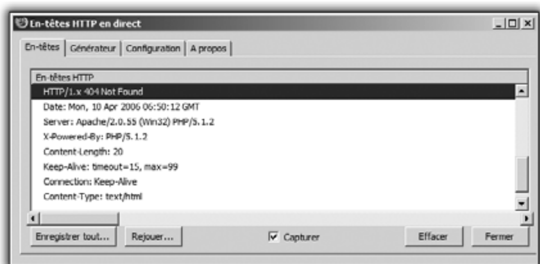


Figure 7.14 : L'en tête indique bien que la page n'existe pas

7.4. Authentification

Le protocole HTTP inclut une gestion d'authentification. Cette authentification met en œuvre le code retour `401`. En recevant ce code retour le navigateur sait qu'il doit ouvrir une petite fenêtre afin de permettre à l'utilisateur de renseigner son identifiant et son mot de passe. En plus du code d'erreur, le serveur doit également préciser le nom de l'espace (`realm`) à protéger avec l'instruction `WWW-Authenticate`.

Une demande d'authentification peut donc être codée de la manière suivante :

Listing 7-8 : Demande d'authentification

```
header('WWW-Authenticate: Basic realm="Zone protégée"');
header("HTTP/1.0 401 Unauthorized");
```

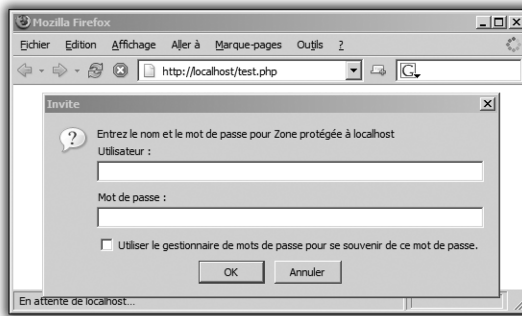


Figure 7.15 : *Firefox propose de vous authentifier dans la "zone protégée"*

Une fois le bouton OK validé, Firefox transmet une requête contenant vos informations d'indentification. Cela se traduit par l'ajout au sein de l'en tête d'une instruction `Authorization` correspondant l'encodage en base 64 de votre identifiant suivi de votre mot de passe :

```
identifiant:mot de passe
```

Vous pouvez vérifier qu'il s'agit bien d'un encodage en base 64 en utilisant la fonction `base64_decode()` sur la valeur associée à `Authorization`.

Listing 7-9 : Cette instruction affiche bien la chaîne "test:test"

```
echo base64_decode("dGVzdDp 0ZXN0");
```

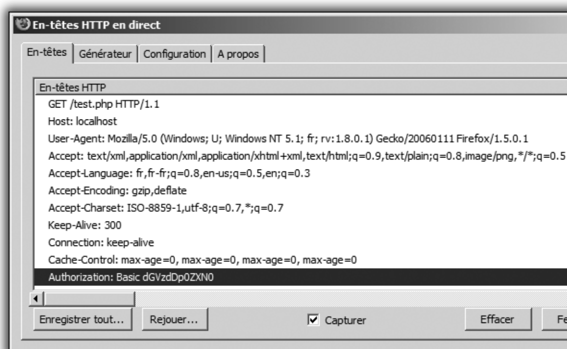


Figure 7.16 : *Transmission encodée de votre identifiant et de votre mot de passe*



Syntaxe

Veillez à bien utiliser un `B` majuscule pour `Basic` et des doubles guillemets pour entourer la valeur associée à `realm`. Dans le cas contraire des problématiques d'incompatibilité avec les navigateurs pourraient survenir.

L'étape suivante consiste à pouvoir vérifier si l'authentification s'est bien déroulée. Les données dont vous disposez pour réaliser cette authentification sont contenues dans la variable super globale `$_SERVER`. Cette variable contient un certain nombre d'informations sur la requête, le script appelé et le serveur web.



Une liste de tous les éléments de $_SERVER est disponible en Annexe.

Dans le cas d'une authentification, `$_SERVER` dispose de deux éléments en plus, `$_SERVER["PHP_AUTH_USER"]` et `$_SERVER["PHP_AUTH_PW"]` qui contiennent respectivement les valeurs que vous avez précisées pour l'identifiant et le mot de passe.

Le principe consiste maintenant à utiliser ces deux valeurs pour vérifier si elles sont correctes. Deux cas apparaissent alors :

- la vérification a échoué et vous proposez à nouveau la demande d'authentification,
- la vérification a réussi et la suite du script est proposée.

Listing 7-10 : Script d'authentification

```
if ($_SERVER["PHP_AUTH_USER"]!="test" ||
    $_SERVER["PHP_AUTH_PW"]!="test") {
  header('WWW-Authenticate: Basic realm="Zone protégée"');
  header("HTTP/1.0 401 Unauthorized");
  exit(0);
}
print("Bienvenue dans la zone sécurisée");
```

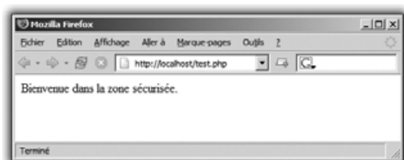


Figure 7.17 : *Message affiché en cas de succès*

En cas de succès les différentes requêtes réalisées avec la même fenêtre de navigateur contiendront l'instruction `Authorization`. Libre à vous de vérifier pour toutes les pages du site que l'internaute est authentifié (en utilisant `$_SERVER["PHP_AUTH_USER"]` et `$_SERVER["PHP_AUTH_PW"]`). Il s'agit d'ailleurs d'une manière classique de protéger l'ensemble des scripts d'un site.

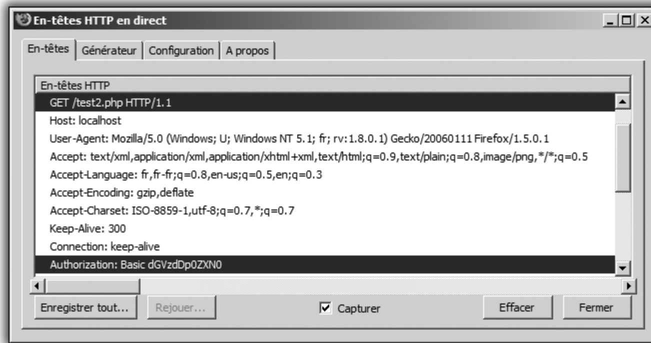


Figure 7.18 : *La requête sur le script test2.php contient bien notre indentification*

Un internaute ne parvenant pas à s'authentifier peut à tout moment cliquer sur le bouton **Annuler** et obtenir un message d'aide. Ce dernier doit suivre l'envoi du code d'erreur `401`.

Listing 7-11 : Précision d'un message d'aide

```
header('WWW-Authenticate: Basic realm="Zone protégée"');
header("HTTP/1.0 401 Unauthorized");
print("Veuillez contacter votre administrateur système ");
print("pour obtenir vos identifiants.");
```

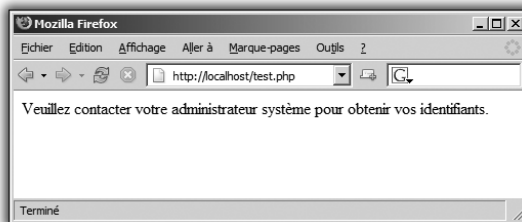


Figure 7.19 : *Message obtenu en cliquant sur Annuler*

L'internaute reste identifié tant que la fenêtre est ouverte. Une porte de sortie doit donc être prévue pour lui permettre de se déconnecter. La technique pour y parvenir consiste à renvoyer un code d'erreur `401`.

L'exemple suivant propose un lien vers un script `quitter.php` chargé de réaliser cette déconnexion.

Listing 7-12 : test.php

```
<?php

if ($_SERVER["PHP_AUTH_USER"]!="test" ||
    $_SERVER["PHP_AUTH_PW"]!="test") {
  header('WWW-Authenticate: Basic realm="Zone protégée"');
  header("HTTP/1.0 401 Unauthorized");
  exit (0);
}
print("Bienvenue dans la zone sécurisée<br/><br/>");
print("<a href='quitter.php'>quitter l'application</a>");

?>
```

Listing 7-13 : quitter.php

```
<?php

header('WWW-Authenticate: Basic realm="Zone protégée"');
header("HTTP/1.0 401 Unauthorized");

?>
```

Une fois identifié, cliquez sur le lien pour basculer sur `quitter.php` et constater que la fenêtre d'authentification est bien proposée. Modifiez maintenant l'url dans la barre de navigation afin d'indiquer **http://localhost/test.php**. Vous constatez à nouveau que la déconnexion a bien fonctionné.

7.5. En bref

- PHP permet de récupérer et de modifier les informations liées aux en-têtes HTTP.
- La fonction `header()` permet d'intervenir sur l'en-tête à la condition expresse de précéder les affichages du script.
- Les codes d'erreurs envoyés par les réponses permettent la gestion des pages d'erreurs et de l'authentification.

Chapitre 8

JavaScript, contrôle de formulaires et AJAX

Présentation de JavaScript 198
Des vérifications simples en PHP 216
Les expressions régulières 222
Ajax 226
Check-list 234

Vous êtes désormais en mesure d'obtenir des informations auprès des internautes et de les transmettre à un script chargé de leur traitement.

L'étape suivante consiste à vérifier que les données du formulaire sont conformes à vos attentes, que tous les champs obligatoires ont été remplis et qu'ils contiennent bien le type de valeur que vous attendez. Suivant la technique utilisée, ces contrôles pourront avoir lieu à la fois au niveau du navigateur (JavaScript) ou du serveur (PHP), avec ou sans rechargement de page.

8.1. Présentation de JavaScript

Le langage JavaScript a été créé en 1995 par Brendan Eich (aujourd'hui employé par Firefox) pour le compte de la société Netscape. Face au succès croissant du Web, Microsoft s'inspira bien vite du concept pour son nouveau navigateur (Internet Explorer) et créa le Jscript en s'éloignant bien évidemment du langage initial. Le besoin évident d'un standard conduisit néanmoins les principaux acteurs à définir une norme : l'ECMAScript.

La situation actuelle est heureusement bien meilleure qu'il y a quelques années. En dehors de Microsoft et de sa mauvaise volonté coutumière, les principaux navigateurs du marché (Firefox, Safari, Opera, Konqueror) tendent à suivre scrupuleusement la norme et permettent d'écrire un code extrêmement portable. Face à ce mouvement, à la croissance rapide de Firefox, et à la grogne générale des développeurs, Microsoft tend cependant à rentrer davantage dans les rangs avec Internet Explorer 7.

En ce qui concerne les nouveautés, JavaScript est un langage qui continue à évoluer et qui a vu un accroissement rapide de sa popularité avec l'avènement du concept Ajax. La version 2.0 de Firefox propose dans ses nouveautés une évolution majeure du langage et son passage à la version 1.7.



JavaScript 2.0

Firefox 3.0 intégrera une version 2 de JavaScript. Cet interpréteur aura la particularité d'avoir été offert par la société Adobe-Macromedia. Il convient en effet de rappeler que le langage action ActionScript utilisé au sein des animations Flash repose à 100% sur l'ECMAScript et qu'il



est par conséquent compatible avec JavaScript. La page **www.mozilla.org/projects/tamarin** propose plus d'informations sur le sujet.

L'objet de ce chapitre n'est certainement pas de rentrer dans les détails du langage. Il s'agit au contraire de présenter quelques fonctionnalités qui pourront se révéler utiles lors de la conception d'applications web. La syntaxe du langage ressemblant énormément à celle du PHP, nous ne nous étendrons pas sur le sujet.

Les fonctions

Comme vous l'avez compris dans le premier chapitre, JavaScript est un langage interprété qui s'exécute au niveau du navigateur. Un code JavaScript se déclare entre les balises `<script type="text/javascript">` et `</script>`. Il se situe le plus souvent dans l'en-tête de la page HTML (entre les balises `<head>` et `</head>`).



Syntaxe alternative

La syntaxe `<script language="javascript">` peut également être trouvée mais appartient désormais au passé.

Un code JavaScript est généralement organisé en fonctions.

Listing 8-1 : Définition de deux fonctions JavaScript : message() et popup()

```
<html>
<head>
<script type="text/javascript">
function message(str)
{
  document.write(str);
}

function popup(str)
{
  alert(str);
}
</script>
</head>
<body>
...
```

Plutôt que placer tout le code JavaScript au niveau de l'en-tête (*header*), vous pouvez le centraliser dans un fichier extérieur.

Listing 8-2 : fichier fonctions.js contenant la déclaration des deux fonctions

```
function message(str)
{
  document.write(str);
}

function popup(str)
{
  alert(str);
}
```

Ce dernier est ensuite inclus dans le fichier HTML de la manière suivante :

Listing 8-3 : Le code JavaScript est inclus depuis un fichier externe

```
<html>
<head>
<script type="text/javascript" src="fonctions.js"></script>
</head>
<body>
...
```

Une fois les fonctions créées et associées à votre page, elles peuvent être appelées de différentes manières : soit directement dans le contenu de la page, soit en les liant à un événement.



Similarités et différences entre PHP et JavaScript

Comme vous le constatez, PHP et JavaScript utilisent la même syntaxe pour déclarer des fonctions. D'autres éléments du langage sont également très proches : les structures de contrôle (`if`, `while`, `for`, etc.), les opérateurs de contrôle (==, >=, etc.), le caractère non typé des variables. Parmi les différences, nous pouvons tout de même citer :

- l'absence du `$` devant les variables ;
- la nécessité de déclarer les variables avant de les utiliser pour la première fois (par exemple, `var n;`) ;
- des noms de fonctions différents (par exemple, `print()` devient `document.write()`) ;
- le signe plus (+) plutôt que le point (`.`) comme opérateur de concaténation de chaînes de caractères.

L'appel direct

Dans le corps de la page (<body>), ouvrez un deuxième bloc de code JavaScript qui vous permet de faire appel directement aux fonctions définies plus haut :

Listing 8-4 : exécution de la fonction message() par appel dans le corps de la page

```
<html>
<head>
<script type="text/javascript">
function message(str)
{
  document.write(str);
}
</script>
</head>
<body>
<script type="text/javascript">
message("message généré par une fonction JavaScript");
</script>
</body>
</html>
```

Comme en PHP, les éléments (fonctions, variables) définis dans les différents blocs de code sont accessibles à partir des autres blocs.

Dans le cas présent, vous auriez pu définir les fonctions juste au-dessus de leur appel :

Listing 8-5 : Déclaration de la fonction et son appel dans le même bloc

```
<html>
<head>
<body>
<script type="text/javascript">
function message(str)
{
  document.write(str);
}
message("message généré par une fonction JavaScript");
</script>
</body>
</html>
```

Afin de faciliter la relecture, l'usage veut cependant de les laisser dans l'en-tête de la page.

Les événements

La nature des événements liés à une page web est assez variée. Un événement peut être lié à la page elle-même (ouverture, fermeture) ou à un widget (clic sur un lien, sur un bouton, focus sur un champ de texte, choix d'une option dans un menu déroulant).

L'exemple ci-dessous ouvre une petite fenêtre (à l'aide de la fonction `alert()`) une fois la page chargée, et une autre quand vous fermez le navigateur. Les deux événements utilisés sont `onLoad()` et `onUnload()` liés à la balise `<body>`.

Listing 8-6 : Exécution de la fonction popup() à l'ouverture et à la fermeture de la page avec des paramètres différents

```
<html>
<head>
<script type="text/javascript">
function popup(str)
{
  alert(str);
}
</script>
</head>
<body onLoad="popup('ouverture de la page')"
      onUnload="popup('fermeture de la page')">
événements JavaScript
</body>
</html>
```

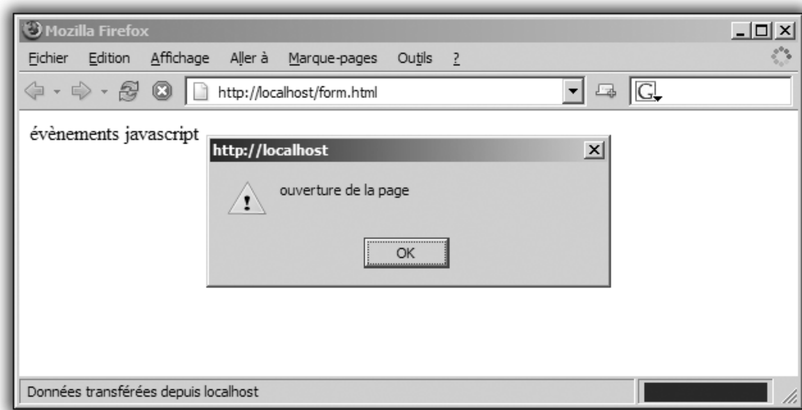


Figure 8.1 : *Apparition de la fenêtre pop-up une fois la page chargée*

Vous allez maintenant déclencher la fonction `popup()` en cliquant sur un bouton :

Listing 8-7 : La fonction popup() est associé au clic sur le bouton

```
<html>
<head>
<script type="text/javascript">
function popup(str)
{
  alert(str);
}
</script>
</head>
<body>
<form>
<input type="button" value="cliquer ici"
       onClick="popup('événement associé au clic')" />
</form>
</body>
</html>
```

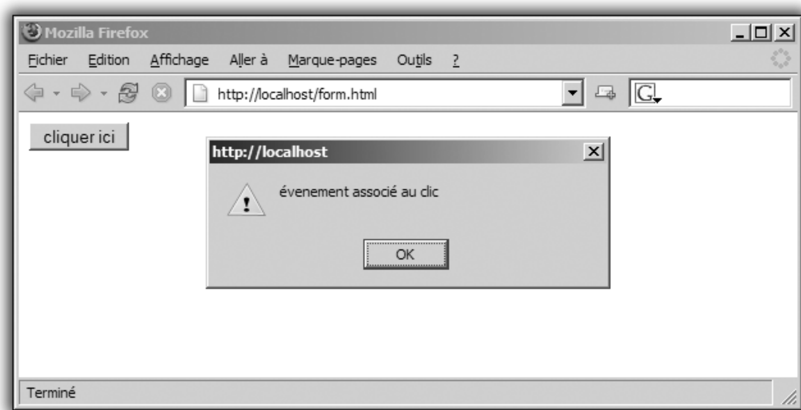


Figure 8.2 : *La fenêtre apparaît à la suite d'un clic sur le bouton*

Plutôt que de créer des fonctions dans l'en-tête, il est préférable, quand celles-ci sont simples, de mettre le code directement au niveau de l'événement.

Listing 8-8 : le code responsable de l'ouverture de la fenêtre pop-up est directement placé au niveau de l'événement

```
<html>
<body>
<form>
<input type="button" value="cliquer ici"
```

```
       onClick="alert('événement associé au clic')" />
</form>
</body>
</html>
```

L'exemple suivant ouvre une fenêtre lorsque vous survolez (événement `onMouseOver()`) un lien :

Listing 8-9 : Affichage d'une fenêtre lorsque le pointeur de la souris passe au-dessus du lien

```
<html>
<body>
<a href="#" onMouseOver="alert('coucou')">lien</a>
</body>
</html>
```

***Tableau 8.1 :** Événements JavaScript*

Événement	Gestionnaire d'événement	Signification
`abort`	`onAbort`	Le chargement d'une image a échoué
`blur`	`onBlur`	Perte du focus d'un élément
`change`	`onChange`	Changement de valeur d'un élément
`click`	`onClick`	Simple clic sur un élément
`dblclick`	`onDblClick`	Double clic sur un élément
`error`	`onError`	Erreur lors du chargement d'une image
`focus`	`onFocus`	Un élément obtient le focus
`keydown`	`onKeyDown`	Une touche du clavier est pressée
`keypress`	`onKeyPress`	Suit le `keydown`
`keyup`	`onKeyUp`	Une touche du clavier est relâchée
`load`	`onLoad`	Lorsque la page est intégralement chargée
`mousedown`	`onMouseDown`	Le bouton de la souris est pressé
`mousemove`	`onMouseMove`	La souris est déplacée
`mouseout`	`onMouseOut`	Le curseur de la souris quitte l'élément

Tableau 8.1 : *Événements JavaScript*

Événement	Gestionnaire d'événement	Signification
`mouseover`	`onMouseOver`	Le curseur de la souris passe sur l'élément
`mouseup`	`onMouseUp`	Le bouton de la souris est relâché
`reset`	`onReset`	Le formulaire est réinitialisé
`resize`	`onResize`	L'élément est redimensionné
`select`	`onSelect`	Le texte est sélectionné
`submit`	`onSubmit`	Lors de la soumission d'un formulaire
`unload`	`onUnload`	Lorsque l'utilisateur quitte la page

L'interaction avec les widgets

JavaScript permet d'interagir avec tous les éléments d'un formulaire. Tout champ peut être initialisé, modifié et contrôlé. Les techniques permettant d'accéder aux valeurs des widgets ont largement évolué depuis 1995. Nous présenterons ici la méthode la plus récente et la plus répandue aujourd'hui.

JavaScript dispose d'une fonction qui vous permet d'accéder à tout élément de la page : `document.getElementById()`. Cette fonction prend en argument l'identifiant de l'élément auquel vous souhaitez accéder. Tout élément d'une page HTML peut être identifié en lui assignant un attribut `id`. La seule contrainte est de choisir un identifiant unique au sein de la page.

Listing 8-10 : ce champ text est désormais identifié en tant que nom

```
<input type="text" id="nom" />
```



Les attributs `name` et `id`

L'attribut `id` ne se substitue en aucun cas à l'attribut `name`. Lors de la soumission d'un formulaire, seul l'attribut `name` est pris en compte pour la transmission des données.

La fonction `document.getElementById()` permet d'accéder à l'ensemble des attributs d'un élément : sa valeur (`value`), son nom (`name`), son identifiant unique (`id`), son type (`type`), etc.

Si vous souhaitez par exemple accéder à la valeur (`value`) d'un `input text` dont l'`id` est `p`, écrivez :

```
document.getElementById('p').value
```

L'accès aux autres attributs de l'élément suit le même principe :

Listing 8-11 : document.getElementById() permet d'accéder à l'ensemble des attributs du widget

```
<html>
<body>

<input type="text" name="prenom" id="p" value="paul" />
<br/><br/>
<input type="button" value="type"
       onClick="alert('type : '+
                document.getElementById('p').type)" />
<br/>
<input type="button" value="name"
       onClick="alert('name : '+
                document.getElementById('p').name)" />
<br/>
<input type="button" value="id"
       onClick="alert('id : '+
                document.getElementById('p').id)" />
<br/>
<input type="button" value="value"
       onClick="alert('value : '+
                document.getElementById('p').value)" />
<br/>

</body>
</html>
```

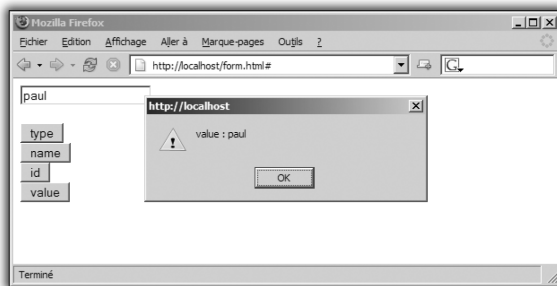


Figure 8.3 : *Chaque bouton permet d'accéder à un attribut*

Les méthodes associées aux éléments peuvent également être appelées. L'exemple suivant illustre l'utilisation de la méthode `select()` qui permet de sélectionner le contenu des `input text` et des `textarea`.

Listing 8-12 : Sélection du contenu de l'input text

```
<html>
<body>

<input type="text" name="prenom" id="p" value="paul" />
<br/><br/>
<input type="button" value="selection du contenu"
      onClick="document.getElementById('p').select()" />
<br/>

</body>
</html>
```

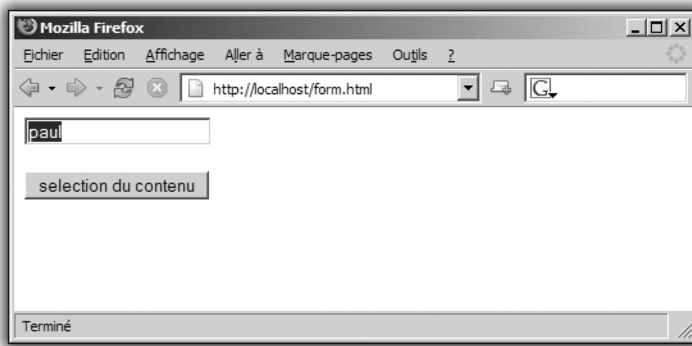


Figure 8.4 : *Sélection du contenu*

La balise `<form>` dispose elle-même d'un certain nombre de méthodes. L'appel direct à sa méthode `submit()` peut éviter de passer par un bouton `submit` pour valider le formulaire.

Listing 8-13 : Le lien situé en dehors du formulaire déclenche l'envoi du formulaire

```
<form id="form1">
…
</form>
<a href="#"
   onClick="document.getElementById('form1').submit()">
```

Présentons avant d'aller plus loin une fonction très utile du JavaScript : `confirm()`. L'appel à cette fonction déclenche l'apparition d'une petite fenêtre (similaire à celle de la fonction `alert()`) qui demande une confirmation (**OK**, **Annuler**) à l'internaute. Si le bouton **OK** est pressé,

`confirm()` retourne le booléen `true`. Associée à une structure de contrôle comme `if`, cette fonction peut permettre d'exécuter différentes parties de code en fonction de la réponse de l'internaute.

Développez à l'aide de cette fonction un petit script qui demande une confirmation avant d'initialiser le contenu d'un champ texte :

Listing 8-14 : Utilisation de la commande confirm()

```
<html>
<header>
<script type="text/javascript">
function ma_fonction()
{
  if (confirm("Initialiser le champ text ?")) {
    document.getElementById('p').value='abcdef';
  }
}
</script>
</header>
<body>
<input type="text" id="p" /><br/><br/>
<input type="button" value="valider"
       onClick="ma_fonction()" /><br/>
</body>
</html>
```

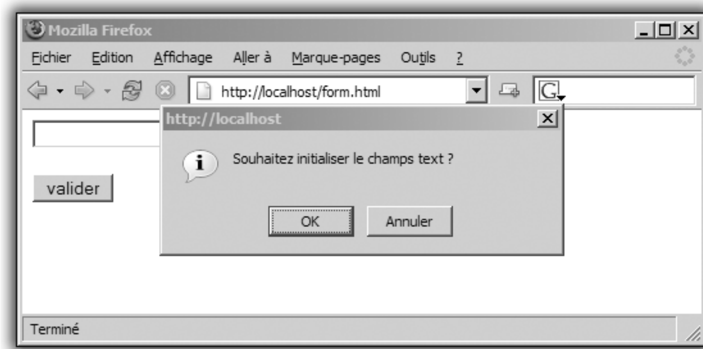


Figure 8.5 : *Utilisation de la fonction confirm*

La fonction `prompt()` permet, quant à elle, de demander durant l'exécution d'un JavaScript une donnée à l'internaute. Faites évoluer le script précédent en initialisant le champ `text` avec une donnée que la fonction `prompt()` aura transmise :

Listing 8-15 : Utilisation de la commande prompt()

```
<html>
<header>
<script type="text/javascript">
function ma_fonction()
{
  if (confirm("Vous souhaitez initialiser le champ text ?")) {
    var str = prompt("Avec quelle valeur ?");
    document.getElementById('p').value=str;
  }
}
</script>
</header>
<body>
<input type="text" id="p" /><br/><br/>
<input type="button" value="valider"
       onClick="ma_fonction()" /><br/>
</body>
</html>
```

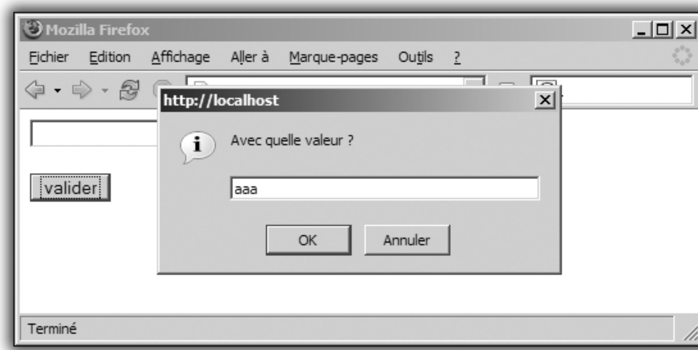


Figure 8.6 : *Utilisation de la fonction prompt*

Nous progressons maintenant vers le but de ce paragraphe : tester les champs d'un formulaire avant de déclencher la transmission.

Il se révèle très facile de tester la présence d'une donnée dans un champ `text`, compte tenu de ce que vous avez vu précédemment :

Listing 8-16 : Vérification du contenu d'un champ text

```
<html>
<header>
<script type="text/javascript">
function verif()
{
  if (document.getElementById('p').value=='') {
```

```
    alert("Erreur");
  }
  else {
    alert("ok");
  }
}
</script>
</header>
<body>
<input type="text" id="p" /><br/><br/>
<input type="button" value="valider"
     onClick="verif()" /><br/>
</body>
</html>
```

La ligne `alert('ok')` aurait pu être ici remplacée par l'appel de la méthode `submit()` du formulaire `monform`.



Débogage JavaScript

Le navigateur Firefox propose un outil très pratique pour les développeurs web : la console JavaScript. Cette console, qui peut être ouverte depuis le menu **Outils**, liste toutes les erreurs qui sont survenues durant l'exécution de votre code et vous donne des indications sur leur origine.



Figure 8.7 : *Une console JavaScript indiquant une mauvaise orthographe de la fonction alert()*

Cette fonctionnalité devrait séduire les développeurs web exaspérés par les messages d'erreur incompréhensibles fournis par Internet Explorer.

Bien que déjà très utile, cette fonctionnalité fait pâle figure face à l'extension Firebug téléchargeable sur le site : **www.getfirebug.com**. Véritable couteau suisse, cette extension permet de tracer les erreurs avec une facilité déconcertante.



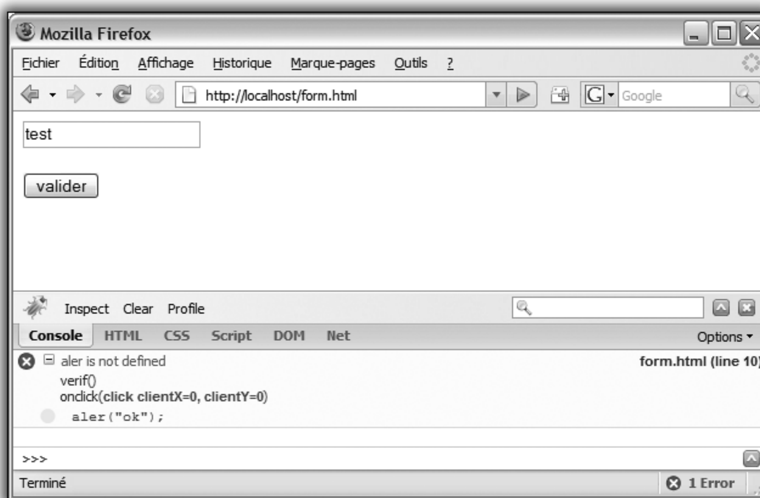


Figure 8.8 : *Affichage de la même erreur dans la console de Firebug*

Les éléments `select`, `input radio` et `input checkbox` doivent être testés différemment.

Les cases à cocher, qu'il s'agisse de `radio` ou de `checkbox`, ne dérogent pas à la règle d'unicité en ce qui concerne l'`id` qui leur est associé. La vérification de l'état « coché » repose sur la valeur (`true`/`false`) de l'attribut `checked`.

```
if (document.getElementById('c').checked==false)
  alert("Cette case n'est pas cochée");
```

Pour les éléments `select`, l'attribut qui nous intéresse est `selectedIndex` qui contient le numéro de l'option sélectionnée (la première a pour index `0`). Ce principe conduit souvent à avoir comme premier élément du `select` une option vide qui force l'internaute à en sélectionner véritablement une. Dans le cas d'un `select multiple`, `selectedIndex` peut avoir la valeur `–1` si aucune des options n'est sélectionnée.

Validez maintenant le formulaire mis en place dans le chapitre précédent. Vous en profiterez pour lui ajouter un champ `annee` que vous limiterez à quatre caractères. Votre objectif est d'afficher un message

d'erreur précisant les champs qui n'ont pas été renseignés. Une fois le formulaire convenablement rempli, une demande de confirmation est faite avant la transmission du formulaire à *script.php*.

```
<html>

<head>
<script type="text/javascript">
function verif()
{
  var err = "";

  if (document.getElementById('idTitre').value=='')
    err = err+"- titre\n";

  if (document.getElementById('idAnnee').value=='')
    err = err+"- année\n";

  if (document.getElementById('idGenre').selectedIndex==-1)
    err = err+"- genre\n";

  if (document.getElementById('idDescript').value=='')
    err = err+"- description\n";

  if (document.getElementById('idCoulOui').checked==false &&
      document.getElementById('idCoulNon').checked==false)
        err = err+"- couleur\n";

  if (document.getElementById('idPays').selectedIndex==0)
    err = err+"- pays\n";

  if (document.getElementById('idStFr').checked==false &&
      document.getElementById('idStGb').checked==false &&
      document.getElementById('idStEs').checked==false)
        err = err+"- sous titre\n";

  if (err!="") {
    alert("Formulaire incomplet :\n"+err);
  }
  else if (confirm("Transmettre le formulaire ?")) {
    document.getElementById('monform').submit();
  }
}
</script>
</head>

<body>

<form action="script.php" id="monform">
```

```
<label>Titre du film</label>
<input type="text" name="titre" id="idTitre" /><br/>

<label>Année</label>
<input type="text" name="annee" id="idAnnee" maxlength="4" />
<br/>

<label>Genre</label>
<select name="genre[]"multiple="yes" size="3" id="idGenre">
 <option value="policier">POLICIER</option>
 <option value="sf">SCIENCE FICTION</option>
 <option value="culte">CULTE</option>
</select><br/>


<label>Description</label>
<textarea name="description" id="idDescript"></textarea><br/>

<label>Film en couleur</label>
<input type="radio" name="couleur" value="1"
       id="idCoulOui" /> oui -
<input type="radio" name="couleur" value="0"
       id="idCoulNon" /> non <br/>

<label>Pays</label>
<select name="pays" id="idPays">
 <option value=""></option>
 <option value="fr">FRANCE</option>
 <option value="us">USA</option>
 <option value="gb">ANGLETERRE</option>
</select><br/>

<label>Sous titre</label>
<input type="checkbox" name="soustitre[]" value="fr"
       id="idStFr" /> français -
<input type="checkbox" name="soustitre[]" value="gb"
       id="idStGb" /> anglais -
<input type="checkbox" name="soustitre[]" value="es"
       id="idStEs" /> espagnol <br/>
<br/>

<input type="button" value="valider" onClick="verif()" />

</form>

</body>
</html>
```

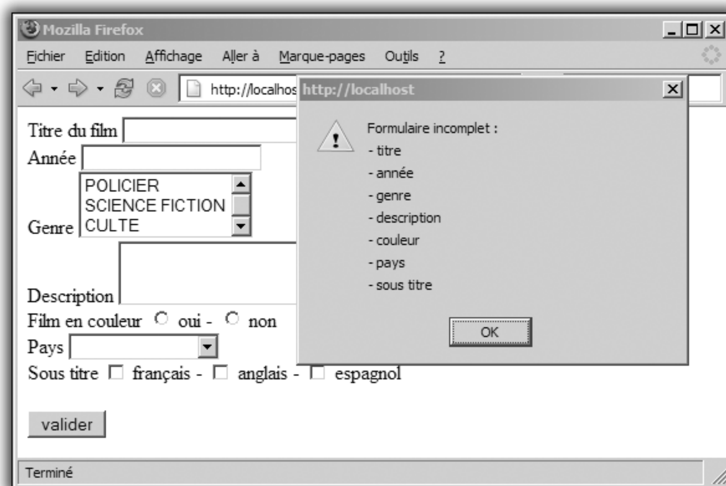


Figure 8.9 : *Validation des différents types de widget*



Les accents

Le choix de l'identifiant `idAnnee` plutôt qu'`idAnnée` est volontaire. La gestion des accents est loin d'être parfaite dans la plupart des navigateurs, et cela risque d'être à l'origine d'erreurs.

La bibliothèque Prototype

Comme vous pouvez le constater, la syntaxe des fonctions JavaScript peut apparaître assez compliquée. Partant de ce constat, certains développeurs liés à la plateforme Ruby On Rails ont conçu une bibliothèque JavaScript visant à simplifier la vie des développeurs et à alléger leur code : Prototype. Cette bibliothèque peut être téléchargée sur le site **www.prototypejs.org**. Pour l'utiliser, placez-la par exemple dans un répertoire `js` contenant vos scripts JavaScript et incluez-la de la manière suivante :

Listing 8-17 : Inclusion de la bibliothèque Prototype

```
<html>
<head>
<script src="/js/prototype.js" type="text/javascript"></script>
</head>
<body>
...
```

Une fois incluse, cette bibliothèque vous permet de remplacer `document.getElementById('id_element')` par `$('id_element')` et `document.getElementById('id_element').value` par `$F('id_element')`. `$()` et `$F()` sont des extensions propres à Prototype vous permettant de rendre votre code beaucoup plus élégant et lisible.

Listing 8-18 : Version Prototype

```
<head>
<script src="/js/prototype.js"
        type="text/javascript"></script>
<script language="javascript">
function verif()
{
  var err = "";
  if ($F('idTitre')=='') err = err+"- titre\n";
  if ($F('idAnnee')=='') err = err+"- année\n";
  if ($('idGenre').selectedIndex==-1)
    err = err+"- genre\n";
  if ($F('idDescript')=='') err = err+"- description\n";
  if ($('idCoulOui').checked==false &&
      $('idCoulNon').checked==false)
        err = err+"- couleur\n";
  if ($('idPays').selectedIndex==0)
    err = err+"- pays\n";
  if ($('idStFr').checked==false &&
      $('idStGb').checked==false &&
      $('idStEs').checked==false)
        err = err+"- sous titre\n";
  if (err!="") {
    return alert("Formulaire incomplet :\n"+err);
  }
  if (confirm("Transmettre le formulaire ?")) {
    $('monform').submit();
  }
}
</script>
</head>
```



Inconvénient de Prototype

L'inconvénient majeur de la bibliothèque Prototype est sa taille. La première fois qu'un internaute visite une page faisant appel à la bibliothèque Prototype, un fichier (*prototype.js*) de 70 Ko doit être téléchargé en plus de la page et des éventuelles images liées. Heureusement pour nous, les navigateurs web sont intelligents et ne téléchargeront pas à nouveau ce fichier à chaque consultation de la page



(ou de toute autre page utilisant la bibliothèque). Le cache du navigateur sera en effet mis à contribution.

8.2. Des vérifications simples en PHP

Les tests JavaScript ne doivent en aucun cas être considérés comme un gage de sécurité absolu. Vous savez en effet qu'il est possible de composer soi-même un lien permettant de transmettre des données. En utilisant cette technique, l'utilisateur passe à travers les tests JavaScript et force la transmission des informations qui lui conviennent. Il est également important de savoir que la technologie JavaScript peut être désactivée au niveau du navigateur. Au bilan, vos vérifications JavaScript doivent plutôt être envisagées comme des avertissements. L'utilisateur est mis au courant que ses données risquent d'être refusées au niveau du script et qu'il lui faudra alors revenir en arrière afin de modifier les données du formulaire. Nous sommes donc ici plus dans une optique de confort, d'ergonomie et de gain de temps.

Commencez par étudier les différentes fonctions de contrôle que vous propose PHP et utilisez le formulaire présenté juste au-dessus pour vos exemples.

Tout d'abord, la fonction `empty()` est de loin la plus importante. Cette fonction retourne `true` si la variable que vous lui avez adressée en argument est vide ou inexistante.

Cette fonction peut être utilisée pour les champs `titre`, `année`, `description` et `pays`.

```
if (empty($_REQUEST['titre'])==true)
{
  print("ERREUR : le champ titre n'a pas été rempli");
  exit();
}
```

Vous savez qu'un `if` vérifie si l'expression est différente de `false`. Vous pouvez donc simplifier votre test et faciliter sa lecture :

```
if (empty($_REQUEST['titre']))
{
  print("ERREUR : le champ titre n'a pas été rempli");
  exit();
}
```



Allégement du code

Il convient de ne pas aller trop loin dans l'allégement de votre code. Une lisibilité accrue pour une relecture facile est largement plus importante qu'un code compact. De plus, n'allez surtout pas imaginer qu'en réduisant la taille de votre code, vous optimiserez sa performance : c'est faux !

La fonction `exit()` est utilisée pour interrompre et quitter définitivement le script. Vous pouvez aussi faire appel à la fonction `die()` qui quitte également le programme tout en affichant à l'écran la chaîne de caractères qui lui est adressée en paramètre.

```
if (empty($_REQUEST['titre']))
{
  die("ERREUR : le champ titre n'a pas été rempli");
}
```



La commande `return ()`

Vous pourrez trouver sur le Web des scripts qui utilisent la fonction `return()` plutôt que les fonctions `exit()` ou `die()`. Cette fonction est pourtant différente dans le sens où `return()` ne quitte que le script en cours. Vous verrez en effet par la suite qu'un script peut en inclure d'autres. Pour être sûr de quitter définitivement le script, il vaut donc mieux utiliser `exit()` ou `die()`.

Ce test n'est cependant pas très fiable car un titre ne contenant qu'un espace serait considéré comme valide. Cette remarque met en exergue l'importance de la préparation et du nettoyage préalables des données avant leur utilisation au sein d'un script. Dans la majorité des cas, il est conseillé de supprimer les caractères dits blancs (espaces, tabulations) situés aux extrémités d'un paramètre. La fonction `trim()` prévue à cet effet rend cette opération triviale.

```
$_REQUEST['titre'] = trim($_REQUEST['titre']);
if (empty($_REQUEST['titre']))
{
  die("ERREUR : le champ titre n'a pas été rempli");
}
```

Trois paramètres restent donc à tester : le genre, la couleur et le sous-titre.

Ces paramètres ont la particularité de ne pas être propagés par le formulaire si aucune valeur n'est cochée ou sélectionnée. Votre test ne porte donc plus sur le caractère vide ou non de la variable, mais sur son existence. La fonction `isset()` vous vient ici en aide en retournant `true` si une variable existe et `false` dans le cas contraire.

```
if (isset($_REQUEST['genre']))
{
  die("ERREUR : aucun genre n'a été sélectionné");
}
```



Les fonctions `empty()` et `isset()`

Même si la fonction `isset()` peut la plupart du temps être remplacée par `empty()`, il est préférable d'être le plus précis possible dans le choix et l'utilisation de vos fonctions.

Vous êtes désormais en mesure de savoir si tous les champs ont été renseignés. Vous n'avez cependant aucune information sur la validité des paramètres reçus. Dans l'état actuel des choses, `$_REQUEST['genre']` pourrait contenir la chaîne de caractères `xx` et passer les tests.

Vous allez donc aller plus loin dans la validation des paramètres et contrôler les points suivants :

- `$_REQUEST['titre']` contient plus de deux caractères ;
- `$_REQUEST['annee']` comprise entre 1930 et 2007 ;
- `$_REQUEST['genre']` contient au moins une des valeurs suivantes : `policier`, `sf`, `culte` ;
- `$_REQUEST['description']` contient entre 10 et 500 caractères ;
- `$_REQUEST['couleur']` vaut `0` ou `1` ;
- `$_REQUEST['pays']` contient une des valeurs suivantes : `fr`, `us`, `gb` ;
- `$_REQUEST['soustitre']` contient au moins une des valeurs suivantes : `fr`, `gb`, `es`.

La fonction `strlen()` qui retourne la taille d'une chaîne de caractères est utilisée pour les tests 1 et 4.

Listing 8-19 : Validation du titre

```
if (strlen($_REQUEST['titre'])>2)
```

Listing 8-20 : Validation de la description

```
if( strlen($_REQUEST['description'])>10 &&
    strlen($_REQUEST['description'])<500)
```

Les tests 2 et 5 peuvent se résumer à des comparaisons de type numérique :

Listing 8-21 : Validation du champ année

```
if ($_REQUEST['annee']>=1930 && $_REQUEST['annee']<=2006)
```

Listing 8-22 : Validation de la couleur

```
if ($_REQUEST['couleur']==0 || $_REQUEST['couleur']==1)
```

Le test 6 se résume quant à lui à des comparaisons sur les chaînes de caractères :

Listing 8-23 : Validation du pays

```
if ($_REQUEST['pays']=='fr' || $_REQUEST['pays']=='us' ||
    $_REQUEST['pays']=='gb')
```

Le principe de ce dernier test est correct, mais pose le problème d'allonger et de complexifier énormément l'`if` dès que vous ajoutez une option. Une technique optimale consiste à déclarer un tableau contenant toutes les valeurs possibles et à vérifier que la valeur de `$_REQUEST['pays']` en fait bien partie. La fonction `in_array()` va vous permettre d'y parvenir. Son premier argument correspond à la valeur à tester et le second à un tableau contenant les valeurs de référence :

Listing 8-24 : Version plus élégante de la vérification du pays

```
$tableau_pays = array('fr','us','gb');
  if (in_array($_REQUEST['pays'],$tableau_pays)==false)
      $erreur .= "- le champ pays est mal rempli<br/>";
```

Les tests 3 et 7 diffèrent du test 6 dans la mesure où `$_REQUEST['genre']` et `$_REQUEST['soustitre']` peuvent contenir plusieurs valeurs qui doivent toutes être valides. La fonction `in_array()` va donc devoir être appliquée pour chacune de ces valeurs :

Listing 8-25 : Validation du genre

```
$tableau_genre = array('policier','sf','culte');
foreach ($_REQUEST['genre'] as $tmp) {
  if (in_array($tmp,$tableau_genre)==false)
      $erreur .= "- le genre $tmp n'est pas correct<br/>";
}
```

Listing 8-26 : Validation du sous-titre

```
$tableau_soustitre = array('fr','gb','es');
foreach ($_REQUEST['soustitre'] as $tmp) {
  if (in_array($tmp,$tableau_soustitre)==false)
      $erreur .= "- le sous-titre $tmp n'est pas
      ✂ correct<br/>";
}
```

Ces deux tests sont valides mais insuffisants car vous ne testez pas le fait que `$_REQUEST['genre']` et `$_REQUEST['soustitre']` sont bien des tableaux et qu'ils contiennent au moins un élément. Cette vérification est importante dans la mesure où `foreach()` génère une erreur lorsque la variable qui lui est adressée en paramètre n'est pas un tableau. Renforcez la vérification en utilisant la fonction `is_array()`, qui indique si la variable qui lui est transmise en paramètre est bien un tableau, et la fonction `count()` qui retourne le nombre d'éléments contenus dans un tableau.

Listing 8-27 : Version plus complète de la validation du genre

```
if (is_array($_REQUEST['genre'])==false ||
    count($_REQUEST['genre'])<=1) {
  $erreur .= "- le genre n'est pas correct<br/>";
}
else {
  $tableau_genre = array('policier','sf','culte');
  foreach ($_REQUEST['genre'] as $tmp) {
    if (in_array($tmp,$tableau_genre)==false)
        $erreur .= "- le genre $tmp n'est pas
        ✂ correct<br/>";
  }
}
```

L'ensemble de ces tests vous informe de la validité des paramètres. Adaptez-les afin de détecter désormais leur non-validité et regroupez-les au sein d'une fonction : `verif()`. Cette fonction retourne `true` si aucune erreur n'est détectée et `false` dans le cas contraire.

Listing 8-28 : Fonction verif() de vérification des paramètres

```
<?php

function verif()
{
  $erreur = "";

  if (strlen($_REQUEST['titre'])<=2)
      $erreur .= "- le champ titre est mal rempli<br/>";
```

```
  if( strlen($_REQUEST['description'])<=10 ||
      strlen($_REQUEST['description'])>=500)
      $erreur .= "- le champ description est mal rempli<br/>";

  if ($_REQUEST['annee']<1930 || $_REQUEST['annee']>2006)
      $erreur .= "- le champ année est mal rempli<br/>";

  if ($_REQUEST['couleur']!=0 && $_REQUEST['couleur']!=1)
      $erreur .= "- le champ couleur est mal rempli<br/>";

  $tableau_pays = array('fr','us','gb');
  if (in_array($_REQUEST['pays'],$tableau_pays)==false)
      $erreur .= "- le champ pays est mal rempli<br/>";

  if (is_array($_REQUEST['genre'])==false ||
      count($_REQUEST['genre'])<1) {
    $erreur .= "- le genre n'est pas correct<br/>";
  }
  else {
    $tableau_genre = array('policier','sf','culte');
    foreach ($_REQUEST['genre'] as $tmp) {
      if (in_array($tmp,$tableau_genre)==false)
          $erreur .= "- le genre $tmp n'est pas correct<br/>";
    }
  }

  if (is_array($_REQUEST['soustitre'])==false ||
      count($_REQUEST['soustitre'])<1) {
    $erreur .= "- le sous-titre n'est pas correct<br/>";
  }
  else {
    $tableau_soustitre = array('fr','gb','es');
    foreach ($_REQUEST['soustitre'] as $tmp) {
      if (in_array($tmp,$tableau_soustitre)==false)
          $erreur .= "- le sous-titre $tmp n'est pas
          ✂ correct<br/>";
      }
  }

  if (!empty($erreur)) {
      print($erreur);
      return false;
  }

  return true;

}

if (verif()==false) exit(0);

print("<b>Titre</b> : ".$_REQUEST['titre']."<br/>");
```

```
print("<b>Année</b> : ".$_REQUEST['annee']."<br/>");
$str_genre = join(',',$_REQUEST['genre']);
print("<b>Genre</b> : ".$str_genre."<br/>");
print("<b>Description</b> : ".$_REQUEST['description']."<br/>");
print("<b>Couleur</b> : ".$_REQUEST['couleur']."<br/>");
print("<b>Pays</b> : ".$_REQUEST['pays']."<br/>");
$str_soustitre = join(',',$_REQUEST['soustitre']);
print("<b>Sous titres</b> : ".$str_soustitre."<br/>");

?>
```

8.3. Les expressions régulières

Les expressions régulières (également appelées *regular expressions* ou *regexp*) permettent de réaliser des tests beaucoup plus fins et complexes. L'idée ici est d'utiliser la fonction `preg_match()` qui retournera `true` si la variable que vous lui adressez en second paramètre « correspond » à l'expression régulière que vous lui avez transmise en premier paramètre (que vous appellerez désormais *pattern*). En terme technique, vous vérifiez si ces deux arguments se correspondent. L'opération générale de recherche de correspondance est appelée le *pattern matching*.



Et le français dans tout ça ?

Comme vous pouvez le constater, il est fait un usage intensif de barbarismes franglais dans cet ouvrage. Le but est avant tout de vous présenter les termes le plus souvent utilisés dans la littérature informatique.

Un pattern est une simple chaîne de caractères entourée par le caractère `/` qui va vous permettre de préciser très finement le motif que vous recherchez dans la variable à tester. Si vous souhaitez par exemple vérifier que la variable `$str` contient quelque part dans son contenu la chaîne `"http"`, écrivez :

Listing 8-29 : Première utilisation de la fonction preg_match()

```
if (preg_match("/http/", $str)) {
  print ("str contient http");
}
else {
  print ("str ne contient pas http");
}
```

Si `$str` contient `"blabla http blabla"`, la fonction `preg_match()` retourne `true`. En revanche, si vous lui adressez `bla ht–tp bla` ou `bla hTTp bla`, `preg_match()` retournera `false`. La fonction `preg_match()` fait en effet partie des fonctions sensibles à la casse (qui font la différence entre les majuscules et les minuscules).

L'option `i` placée derrière le deuxième `/` permet de réaliser une comparaison de type *case insensitive* (qui n'est pas sensible à la casse).

Listing 8-30 : Test insensible à la casse

```
if (preg_match("/a/i",$str)) {
  print("str contient  a ou A");
}
else {
  print("str ne contient pas a ou A");
}
```

Pour vérifier que `"http"` se trouve en début de chaîne, vous disposez de l'accent circonflexe (`^`) qui, dans le cadre d'une regexp, correspond à un début de ligne.

Listing 8-31 : Première utilisation de la fonction preg_match()

```
if (preg_match("/^http/",$str)) {
  print("str commence http");
}
else {
  print("str ne commence pas http");
}
```

Dans le même esprit, le caractère `$` correspond à une fin de ligne. Le pattern `^http$` permet par conséquent de vérifier que `$str` contient exactement la chaîne `"http"`. Dans un tel cas, le test `if ($str=="http")` reste cependant largement plus rapide et pertinent qu'`if (preg_match("/^http$/",$str))`.



Expressions régulières et norme

Les expressions régulières sont aujourd'hui largement normalisées. Leur syntaxe est quasi la même qu'en C, Perl, etc.

Le point (`.`) a aussi un rôle spécial au sein d'un pattern, il correspond à un (et un seul) caractère, quel qu'il soit.

- `preg_match("/ht.tp/","htatp")` retourne `true` ;

- `preg_match("/ht.tp/","htap")` retourne `false`.

Le pattern `^..$` est donc un moyen de tester que `$str` contient deux caractères. Le test `if (strlen($str)==2)` est cependant plus optimisé pour ce type de vérification.

Pour vérifier que `$str` contient bien le point, vous devez le « protéger » avec la barre oblique inversée (\). Le pattern `www\.` permet de vérifier que `$str` contient " `www.`" et non " `www#`". La barre oblique inversée doit également être utilisée pour protéger les caractères suivants : `' / ^ [ ] $ ( ) | * { } + ? { \ `'.

Ces caractères vont en effet permettre d'aller plus loin dans la construction des patterns. Les crochets par exemple permettent de regrouper un ensemble de caractères qui devront apparaître au moins une fois dans `$str`.

- `preg_match("[aeiouy]",$str)` vérifie que `$str` contient au moins une voyelle ;
- `preg_match("[aeiouy]r",$str)` vérifie que `$str` contient au moins une fois une voyelle suivie de la lettre *r* ;

Certains regroupements de caractères peuvent être simplifiés à l'aide des expressions spéciales :

- `\w` : pour des lettres, des chiffres ou le caractère _ ;
- `\d` : pour des chiffres ;
- `\s` : pour des caractères d'espacement : `\n, ,\t,`.

Le pattern `^\d\d\s\d\d\s\d\d\s\d\d\s\d\d$` peut être utilisé pour vérifier que `$str` correspond à un numéro de téléphone (ex: `01 53 98 73 40`).

Vous pouvez une nouvelle fois simplifier ce pattern en utilisant les expressions de fréquence suivantes :

- `"(to)?"` : la chaîne contient une fois au maximum la chaîne `"to"` ;
- `"(to)+"` : la chaîne contient une fois au minimum la chaîne `"to"` ;
- `"(to)*"` : la chaîne contient zéro ou plusieurs fois la chaîne `"to"` ;

- `"(to){2}"` : la chaîne contient deux chaînes `"to"` qui se suivent ;
- `"(to){2,5}"` : la chaîne contient entre deux et cinq chaînes `"to"` qui se suivent.

Votre pattern « téléphonique » devient donc `^(\d\d\s){4}(\d\d)$`.

Voyez quelques exemples :

- `"^\w+$"` pour vérifier que `$str` ne contient que des lettres ;
- `"^[\w\s,\.]+$"` pour vérifier que `$str` n'est composée que de lettres, de chiffres, de caractères blancs ou de points et de virgules.

Les crochets permettent également de définir des intervalles :

- `[a-z]` pour des lettres minuscules ;
- `[A-Z]` pour des lettres majuscules ;
- `[0-9]` pour des chiffres de 0 à 9 ;
- `[0-5]` pour des chiffres de 0 à 5 ;
- `[d-g]` pour des lettres minuscules de *d* à *g*.

Le caractère `|` au sein d'un pattern prend la signification d'un OU :

- `"(fr|gb|es)"` pour vérifier que la chaîne contient `fr` ou `gb` ou `es` ;
- `"^(fr|gb|es)$"` pour vérifier qu'elle est égale à `fr`, `gb` ou `es`.

Compilez maintenant vos connaissances afin de trouver le pattern qui vous permettra de tester qu'une chaîne est une adresse de courriel valide. Pour simplifier et ne pas vous perdre dans des détails, considérez qu'une adresse de courriel est construite de la manière suivante : le nom de l'utilisateur (caractères alphanumériques ainsi que les caractères `- .`), l'arobase (`@`), le nom de domaine (caractères alphanumériques ainsi que les caractères `- .`), un point et l'extension du domaine (de deux à trois caractères).

- étape 1 : `^[\w\.-]+` ;
- étape 2 : `@` ;
- étape 3 : `[\w\.-]+` ;
- étape 4 : `\.` ;

- étape 5 : `[a-z]{2,3}$^`.

Listing 8-32 : Validation d'une adresse de courriel

```
if (preg_match("/^[\w\.-]+@[\w\.-]+\.[a-z]{2,3}$/i",
               $str))
{
  print("Cette adresse e-mail est valide.");
}
```



Et pourquoi pas `ereg()` ?

PHP propose effectivement une autre fonction permettant de manipuler les expressions régulières : `ereg()` (ou `eregi()` pour ignorer la casse). Quoique fonctionnant de façon satisfaisante, `ereg()` souffre de l'inconvénient majeur d'avoir une « espérance de vie » désormais assez courte. Les développeurs de PHP ont en effet décidé pour PHP6 de ne plus supporter `ereg()` et de reporter tous leurs efforts sur la famille des fonctions `preg`. Les fonctions `preg` étant également plus rapides et compatibles avec les données binaires, n'hésitez surtout pas à oublier `ereg` !

8.4. Ajax

Abréviation d'Asynchronous JavaScript and XML, Ajax englobe un certain nombre de technologies qui, pour schématiser, permettent de s'affranchir du schéma classique : une action implique un rechargement de page. L'objet JavaScript `XMLHttpRequest` qui est au cœur d'Ajax permet en effet de transmettre des données au serveur et de récupérer le résultat. En traitant ce résultat et en s'appuyant sur les technologies DHTML/CSS, des interfaces graphiques avancées peuvent être mises en œuvre pour le plus grand confort de l'internaute. L'outil cartographique de Google (**http://maps.google.com**) est un des plus beaux exemples d'environnement Ajax.

AJAX et Prototype

L'objet `XMLHttpRequest`, disponible maintenant dans tous les navigateurs, n'est hélas pas standardisé au niveau de son utilisation. Une nouvelle fois, la bibliothèque Prototype nous vient en aide en masquant toutes les difficultés liées aux différentes méthodes d'initialisation.

Notre premier exemple consiste à transmettre une opération au script calculatrice.php qui calculera le résultat et le renverra à notre script, tout ceci sans quitter la page !

Prototype rend cette opération extrêmement aisée avec la méthode request() qui étend les fonctionnalités d'un élément de type formulaire. Cette méthode prend en paramètre un objet dont le rôle est de définir les comportements de l'application en cas de succès ou d'échec.

Un objet JavaScript se construit de la façon suivante :

Listing 8-33 : Création et utilisation de l'objet rectangle

```
<script>
var rectangle = {
  longueur:3,
  largeur:2,
  surface:function () {
    return this.longueur * this.largeur;
  }
};
alert('surface='+rectangle.surface());
</script>
```

La méthode request() va automatiquement récupérer le script à appeler en allant lire la valeur de l'attribut action du formulaire. Les paramètres transmis correspondent aux champs du formulaire.

Listing 8-34 : Formulaire permettant de définir l'opération : calculatrice.html

```
<html>
<header>
<script src="/js/prototype.js"
✂ type="text/javascript"></script>
<script type="text/javascript">
function calcule()
{
  if ($F('a').blank() || $F('b').blank()) {
    alert('Veuillez réviser les valeurs.');
    return;
  }
  $('calculatrice').request({
    onComplete: function(transport){
      alert(transport.responseText)
    }
  });
}
```

```
</script>
</header>
<body>
<form id="calculatrice" method="get"
      action="calculatrice.php">
  <input type="text" name="a" id="a" />
  <select name="operation">
    <option value="addition"> + </option>
    <option value="soustraction"> - </option>
    <option value="multiplication"> * </option>
    <option value="division"> / </option>
  </select>
  <input type="text" name="b" id="b" />
  <input type="button" value="calculer"
         onClick="calcule()" /><br/>
</form>
</body>
</html>
```

Pour retourner une valeur, le script PHP se contente d'écrire le résultat sur la sortie standard.

Listing 8-35 : Script calculant le résultat de l'opération : calculatrice.php

```
<?php

$a = $_REQUEST['a'];
$b = $_REQUEST['b'];

switch ($_REQUEST['operation']) {
  case 'addition':
    $resultat = $a + $b;
    break;
  case 'soustraction':
    $resultat = $a - $b;
    break;
  case 'multiplication':
    $resultat = $a * $b;
    break;
  case 'division':
    $resultat = $a / $b;
    break;
}

print ("Resultat : ".$resultat);
?>
```

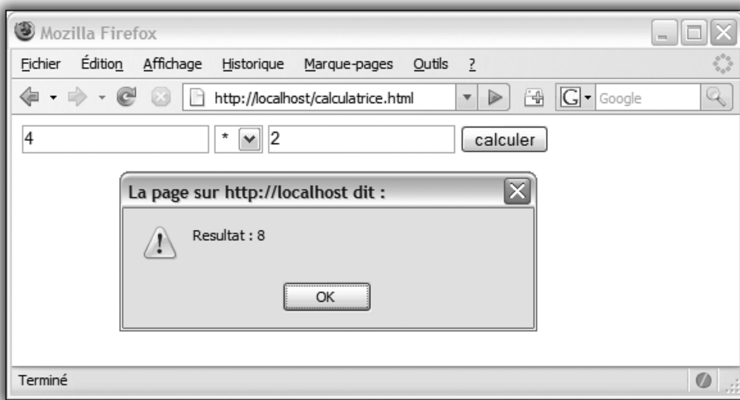


Figure 8.10 : *Le résultat apparaît sans rechargement de la page*

Échange de données au format JSON

Vous savez désormais échanger une donnée entre un script et le serveur. L'idéal serait cependant de pouvoir transmettre plusieurs informations dans le cadre d'une seule et même transaction. Il conviendrait pour cela que le script PHP puisse transmettre un tableau de valeurs à la fonction JavaScript. Par défaut cependant, un tableau PHP et un tableau JavaScript n'ont absolument rien à voir.

Heureusement pour nous, le format JSON rend possible l'échange de données complexes (tableaux, objets) entre JavaScript et PHP (et inversement). La *JavaScript Object Notation* a le triple avantage d'être légère, lisible et interprétable par un très grand nombre de langages de haut niveau (Ruby, Java, C#, Python, etc.).

Un échange JSON typique se déroule ainsi :

- une donnée PHP est convertie au format JSON ;
- cette chaîne de caractères représentant la donnée PHP est transmise au JavaScript ;
- le script décrypte la chaîne et la convertit en sa représentation JavaScript.

Une opération similaire pourrait tout à fait avoir lieu dans le sens inverse : de JavaScript vers PHP.

Les fonctions de manipulation de données au format JSON sont incluses à la fois dans PHP 5.2 et dans la bibliothèque Prototype (1.5).

Tableau 8.2 : *Gestion du format JSON*

	PHP (5.2)	JavaScript (Prototype)
Encodage d'une variable au format JSON	`json_encode ($variable)`	`Object.toJSON(var)`
Décodage d'une variable au format JSON	`json_decode ($json_string)`	`json_string .evalJSON()`

L'exemple suivant permet de valider un formulaire en restant sur la même page tout en passant par le serveur pour réaliser les vérifications.

En cliquant sur le bouton du formulaire, la fonction JavaScript `verifAJAX()` envoie une requête AJAX au script `verif.php` qui contrôle les champs et retourne un objet au format JSON dont chaque attribut correspond à un champ invalide. La fonction JavaScript `verifAJAX()` reçoit la chaîne de caractères correspondant à l'objet, le convertit en objet JavaScript et affiche un message d'erreur si l'objet contient des attributs, ou un message de félicitations dans le cas contraire.

Listing 8-36 : Formulaire

```
<html>
<head>
<script src="/js/prototype.js" type="text/javascript"></script>
<script type="text/javascript">
function verifAJAX()
{
  $('monform').request({
    onComplete: function(transport){
      var errors = transport.responseText.evalJSON();
      var message = "";
      for (var id in errors) {
        message += "\n - "+errors[id];
      }
      if (message=="") {
        alert("Félicitations !");
      }
      else {
        alert("Veuillez vérifier les champs suivants : "+
              message);
```

```
            }
        }
    });
}
</script>
</head>
<body>

<form action="verif.php" id="monform">

<label>Titre du film</label>
<input type="text" name="titre" id="idTitre" /><br/>

<label>Année</label>
<input type="text" name="annee" id="idAnnee" maxlength="4" />
<br/>

<label>Genre</label>
<select name="genre[]"multiple="yes" size="3" id="idGenre">
 <option value="policier">POLICIER</option>
 <option value="sf">SCIENCE FICTION</option>
 <option value="culte">CULTE</option>
</select><br/>


<label>Description</label>
<textarea name="description" id="idDescript"></textarea><br/>

<label>Film en couleur</label>
<input type="radio" name="couleur" value="1"
       id="idCoulOui" /> oui -
<input type="radio" name="couleur" value="0"
       id="idCoulNon" /> non <br/>

<label>Pays</label>
<select name="pays" id="idPays">
 <option value=""></option>
 <option value="fr">FRANCE</option>
 <option value="us">USA</option>
 <option value="gb">ANGLETERRE</option>
</select><br/>

<label>Sous titre</label>
<input type="checkbox" name="soustitre[]" value="fr"
       id="idStFr" /> français -
<input type="checkbox" name="soustitre[]" value="gb"
       id="idStGb" /> anglais -
<input type="checkbox" name="soustitre[]" value="es"
       id="idStEs" /> espagnol <br/>
<br/>
```

```
<input type="button" value="valider" onClick="verifAJAX()" />

</form>

</body>
</html>
```



Inclusion de la bibliothèque Prototype

N’oubliez pas d’inclure la bibliothèque Prototype avec la balise `script` située dans le header de la page.

Listing 8-37 : Script de validation

```
<?php

function verif()
{
  $ret_arr = array();

  if (strlen($_REQUEST['titre'])<=2) {
    $ret_arr[] = 'titre';
  }

  if( strlen($_REQUEST['description'])<=10 ||
      strlen($_REQUEST['description'])>=500) {
    $ret_arr[] = 'description';
  }

  if ($_REQUEST['annee']<1930 || $_REQUEST['annee']>2006) {
    $ret_arr[] = 'annee';
  }

  if (!array_key_exists('couleur', $_REQUEST)) {
    $ret_arr[] = 'couleur';
  }

  $tableau_pays = array('fr','us','gb');
  if (in_array($_REQUEST['pays'],$tableau_pays)==false) {
    $ret_arr[] = 'pays';
  }

  if (is_array($_REQUEST['genre'])==false ||
      count($_REQUEST['genre'])<1) {
    $ret_arr[] = 'genre';
  }
  else {
    $tableau_genre = array('policier','sf','culte');
    foreach ($_REQUEST['genre'] as $tmp) {
```

```
        if (in_array($tmp,$tableau_genre)==false) {
          $ret_arr[] = 'genre';
          break;
        }
      }
    }

    if (is_array($_REQUEST['soustitre'])==false ||
        count($_REQUEST['soustitre'])<1) {
      $ret_arr[] = 'soustitre';
    }
    else {
      $tableau_soustitre = array('fr','gb','es');
      foreach ($_REQUEST['soustitre'] as $tmp) {
        if (in_array($tmp,$tableau_soustitre)==false) {
          $ret_arr[] = 'soustitre';
          break;
        }
      }
    }

    return (object)$ret_arr;
  }

  print (json_encode(verif()));

  ?>
```

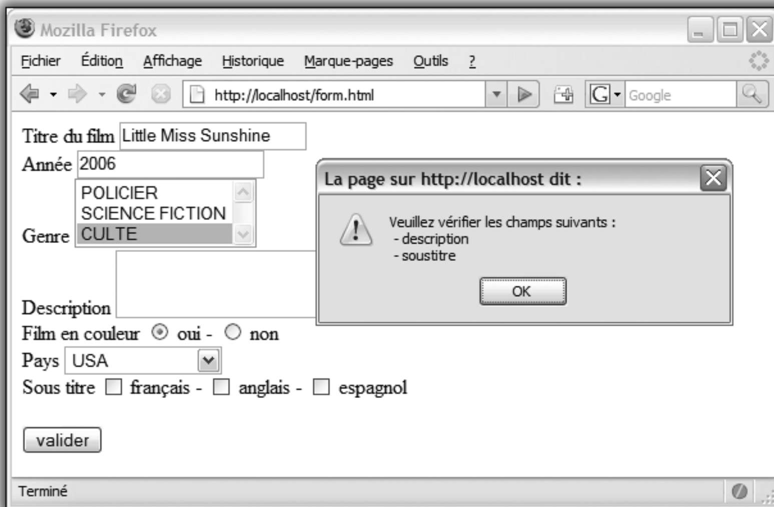


Figure 8.11 : *Les vérifications proviennent du serveur*

Ce mode de fonctionnement dispose des avantages suivants :

- vous ne dupliquez pas les tests à la fois en JavaScript et en PHP ;
- la validation au niveau serveur est à la fois plus sûre et plus aisée ;
- l'internaute ne quittant pas la page, le contenu du formulaire n'est pas perdu.

Dans une utilisation normale, le script PHP réaliserait une action supplémentaire dans le cas où aucune erreur ne serait détectée. Les données transmises pourraient être envoyées par mail ou insérées dans une base de données.

8.5. Check-list

- Les JavaScripts sont exécutés au niveau du navigateur.
- L'exécution d'un JavaScript fait suite à un événement sur la page web.
- Les JavaScripts permettent de vérifier si le formulaire a bien été rempli.
- Les données reçues par un script PHP doivent toujours être contrôlées avant d'être exploitées.
- Les contrôles JavaScript ne suffisent pas pour une validation sérieuse. Des contrôles PHP doivent systématiquement venir les compléter.
- Les expressions régulières permettent de vérifier très précisément qu'une variable correspond à un modèle donné (*pattern*).
- AJAX permet au navigateur web de communiquer avec le serveur sans avoir à recharger une nouvelle page.

Chapitre 9

L’envoi d’un formulaire par courriel

Configuration requise 236
Mail Texte 237
Mail HTML 242
Check-list 248

L’envoi par courriel d’informations en provenance d’un formulaire est certainement l’utilisation la plus répandue de PHP. Vous verrez dans ce chapitre que PHP vous facilite largement la tâche pour les envois les plus simples. Vous constaterez en revanche que les courriels mis en forme (HTML) nécessitent davantage de connaissances, notamment en ce qui concerne le standard MIME.

9.1. Configuration requise

Alors que l’envoi de courriel ne nécessite aucune configuration particulière si vos scripts sont exécutés chez un hébergeur, il en va tout autrement s’ils sont placés sur votre machine. Arrêtons-nous un instant sur les adaptations à apporter à votre environnement de travail pour le rendre compatible avec l’envoi de courriels.



Hébergements gratuits

Certains hébergeurs gratuits interdisent l’usage de la commande `mail()` afin d’éviter les abus de type spam. N’hésitez donc pas, avant de choisir un hébergeur, à vous renseigner sur l’étendue des limitations au niveau des fonctionnalités du langage.

Si votre version de WampServer n’est pas suffisamment récente, vous pourrez découvrir, en parcourant le fichier *php.ini*, la section suivante :

Listing 9-1 : section consacrée à l’envoi de courriels dans le fichier php.ini

```
[mail function]
; For Win32 only.
SMTP = localhost

; For Win32 only.
;sendmail_from = me@example.com
```

La ligne `SMTP = localhost` indique que PHP est paramétré pour utiliser votre propre machine (`localhost`) pour l’envoi des courriels. Il s’agit là de la directive de configuration par défaut qui est loin de convenir.

Le serveur que vous devez utiliser est celui qui est proposé par votre fournisseur d’accès. Il s’agit du serveur SMTP par lequel vous passez également dans votre gestionnaire de courriels (par exemple, Outlook Express, Thunderbird, etc.). Chez Free, le serveur SMTP a pour adresse

smtp.free.fr. Cette norme est à peu près respectée par l'ensemble des FAI (fournisseurs d'accès à Internet). Nous pouvons citer à titre d'exemple : **smtp.wanadoo.fr**, **smtp.noos.fr**, **smtp.club-internet.fr**, etc.

Intéressons-nous maintenant à la ligne `;sendmail_from = me@example.com`. Le point-virgule initial signifie qu'elle est commentée et qu'elle n'est donc pas prise en compte. Cette directive permet de préciser l'origine des courriels envoyés depuis votre machine. Cette directive est importante dans la mesure où les serveurs de courriels « relais » refuseront de faire suivre votre message si son origine n'est pas précisée. Veillez donc à supprimer le point-virgule et à renseigner votre adresse e-mail.

Une fois ces directives de configuration modifiées, le serveur Apache doit être redémarré.



Accès restreints

Les FAI n'autorisent que leurs clients à utiliser leur serveur SMTP. Un client Wanadoo ne pourra en aucun cas passer par **smtp.free.fr**. Cette situation peut se révéler pénible si vous travaillez sur un portable et que le fournisseur d'accès change d'un lieu à un autre. Une solution consiste à passer par un serveur SMTP gratuit qui vous autorisera à transmettre des courriels quelle que soit votre connexion à Internet. Google propose désormais ce service à l'ensemble de ses clients Gmail (**www.gmail.com**).

9.2. Mail Texte

Comme vous l'avez vu en introduction, l'envoi de courriels en PHP est simple. Il suffit d'utiliser la fonction `mail()`.

Les arguments de cette fonction sont :

- l'adresse de destination ;
- le titre du message ;
- le contenu du message ;
- d'éventuelles options.

La fonction `mail()` est généralement appelée de la manière suivante :

```
mail($destinataire,$titre,$message);
```

- `$email` est l'e-mail de la personne qui va recevoir le courriel, par exemple `$destinataire = "bill@domaine.fr";`.
- `$titre` est le titre de l'e-mail, par exemple `$titre = "réponse au formulaire";`.
- `$message` est le corps du message qui va contenir toutes les informations.

La fonction `mail()` retourne un booléen qui indique si l'envoi s'est bien déroulé. Ce statut ne concerne que l'envoi, il n'indique en aucun cas le fait que le courriel est bien arrivé dans la boîte du destinataire.

Commencez par construire ce message :

Listing 9-2 : Construction du contenu du courriel

```
$message = "";
$message .= "Titre : ".$_REQUEST['titre']."\n";
$str_genre = join(',',$_REQUEST['genre']);
$message .= "Genre : ".$str_genre."\n");
$message .= "Description : ".$_REQUEST['description']."\n");
$message .= "Couleur : ".$_REQUEST['couleur']."\n");
$message .= "Pays : ".$_REQUEST['pays']."\n");
$str_soustitre = join(',',$_REQUEST['soustitre']);
$message .= "Sous titres : ".$str_soustitre."\n";
```

La première ligne initialise la variable (=), les suivantes ajoutent des informations à la fin de la variable (`.=`).

Vous remarquez l'usage systématique du caractère `\n` à la fin de chaque ligne. Il s'agit du caractère représentant un saut de ligne.

Jusqu'à maintenant, vous utilisiez la balise `<BR/>` pour effectuer des sauts de ligne car les informations générées par vos scripts s'affichaient dans un navigateur web. Or, en HTML, un saut de ligne est représenté par la balise `<BR/>`.

Dans le cas présent, c'est un lecteur de courriels (de type Thunderbird ou Outlook) qui va afficher l'information. Cette fois, les données sont considérées comme du texte brut. Le caractère de saut de ligne est alors représenté par un `\n` (ou `\r\n` sous Windows). Il existe d'autres caractères spéciaux qui peuvent être utilisés dans du texte brut, par exemple le `\t` qui correspond à une tabulation.

Votre script prend finalement la forme suivante :

```
<?php

function verif()
{
  $erreur = "";

  if (strlen($_REQUEST['titre'])<=2)
      $erreur .= "- le champ titre est mal rempli<br/>";

  if( strlen($_REQUEST['description'])<=10 ||
      strlen($_REQUEST['description'])>=500)
      $erreur .= "- le champ description est mal rempli<br/>";

  if ($_REQUEST['annee']<1930 || $_REQUEST['annee']>2006)
      $erreur .= "- le champ année est mal rempli<br/>";

  if ($_REQUEST['couleur']!=0 && $_REQUEST['couleur']!=1)
      $erreur .= "- le champ couleur est mal rempli<br/>";

  $tableau_pays = array('fr','us','gb');
  if (in_array($_REQUEST['pays'],$tableau_pays)==false)
      $erreur .= "- le champ pays est mal rempli<br/>";

  if (is_array($_REQUEST['genre'])==false ||
      count($_REQUEST['genre'])<1) {
    $erreur .= "- le genre n'est pas correct<br/>";
  }
  else {
    $tableau_genre = array('policier','sf','culte');
    foreach ($_REQUEST['genre'] as $tmp) {
      if (in_array($tmp,$tableau_genre)==false)
          $erreur .= "- le genre $tmp n'est pas correct<br/>";
    }
  }

  if (is_array($_REQUEST['soustitre'])==false ||
      count($_REQUEST['soustitre'])<1) {
    $erreur .= "- le sous-titre n'est pas correct<br/>";
  }
  else {
    $tableau_soustitre = array('fr','gb','es');
    foreach ($_REQUEST['soustitre'] as $tmp) {
      if (in_array($tmp,$tableau_soustitre)==false)
          $erreur .= "- le sous-titre $tmp n'est pas
          ✂ correct<br/>";
      }
  }

  if (!empty($erreur)) {
```

```
      print($erreur);
      return false;
    }

  return true;

}

if (verif()==false) exit(0);

print("<b>Titre</b> : ".$_REQUEST['titre']."<br/>");
print("<b>Année</b> : ".$_REQUEST['annee']."<br/>");
$str_genre = join(',',$_REQUEST['genre']);
print("<b>Genre</b> : ".$str_genre."<br/>");
print("<b>Description</b> : ".$_REQUEST['description']."<br/>");
print("<b>Couleur</b> : ".$_REQUEST['couleur']."<br/>");
print("<b>Pays</b> : ".$_REQUEST['pays']."<br/>");
$str_soustitre = join(',',$_REQUEST['soustitre']);
print("<b>Sous titres</b> : ".$str_soustitre."<br/>");

$destinataire = "fx@kernix.com";
$titre = "réponse au formulaire";

$message = "";
$message .= "Titre : ".$_REQUEST['titre']."\n";
$message .= "Année : ".$_REQUEST['annee']."\n";
$str_genre = join(',',$_REQUEST['genre']);
$message .= "Genre : ".$str_genre."\n";
$message .= "Description : ".$_REQUEST['description']."\n";
$message .= "Couleur : ".$_REQUEST['couleur']."\n";
$message .= "Pays : ".$_REQUEST['pays']."\n";
$str_soustitre = join(',',$_REQUEST['soustitre']);
$message .= "Sous titres : ".$str_soustitre."\n";

if (mail($destinataire,$titre,$message)==true) {
  print("<hr/>Les informations ont bien été transmises.");
}
else {
  die("<hr/>L'envoi du courriel a échoué.");
}

?>
```

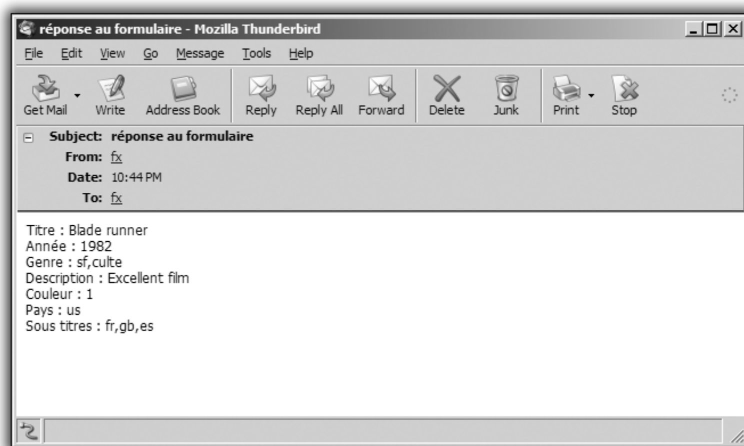


Figure 9.1 : *Message envoyé par le script et visualisé avec Thunderbird*



Présentation des courriels en mode texte

Certains gestionnaires de courriels reconnaissent en mode texte des balises qui permettent d'enrichir visuellement le contenu. Un mot peut ainsi être passé en gras s'il est entouré d'astérisques (*), en souligné avec d _ et en italique avec /.

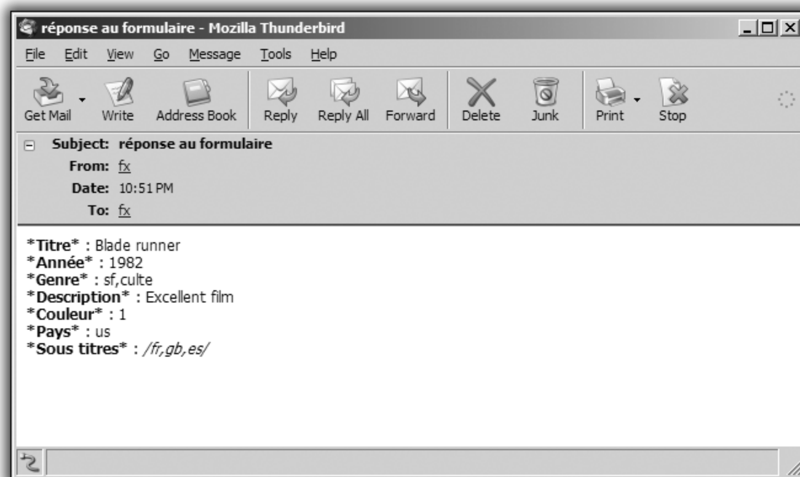


Figure 9.2 : *Mise en forme minimale dans un courriel en mode texte*

9.3. Mail HTML

Le script que vous venez d’écrire ne permet d’envoyer un courriel qu’au format texte.

Mettre une page HTML dans le corps du message ne fonctionnerait pas. L’émission d’un courriel au format HTML nécessite la mise en œuvre du quatrième paramètre de la fonction `mail()`. Ce paramètre est une chaîne de caractères contenant des informations qui seront ajoutées à l’en-tête (*header*) du message.

Un courriel est composé de deux parties principales : l’en-tête et le corps du message. L’en-tête contient un certain nombre d’informations sur le courriel : son origine, le destinataire, le format, le sujet, l’heure d’envoi, le logiciel d’envoi.

L’organisation des données dans cet en-tête est très simple :

```
champs1: valeur1
champs2: valeur2
etc.
```

Voyez cet exemple d’en-tête de courriel :

```
From: michel@toto.fr
To: Paul@titi.fr
Subject: retour de vacances
X-Mailer: Microsoft Outlook Express
```

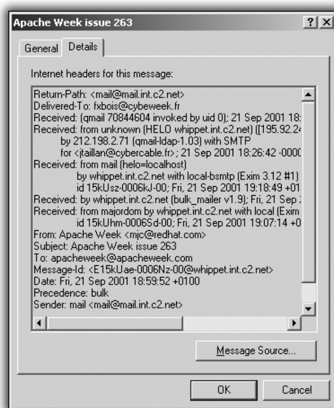


Figure 9.3 :
Partie de l’en-tête d’un courriel (vu avec Outlook Express)

La commande `mail()` compose donc un en-tête par défaut en intégrant les données passées en paramètres : le destinataire (`To:`), le sujet

(Subject:). Le quatrième paramètre permet d'ajouter certaines informations à cet en-tête.

Il est courant qu'une personne recevant un courriel émanant d'un script PHP ne sache pas qui lui a envoyé. En ajoutant "From: michel@toto.fr" comme quatrième paramètre, le destinataire est en mesure de connaître l'origine du courriel.

Les lignes contenues dans le quatrième paramètre doivent être séparées par des sauts de ligne : \n.

Ajoutez également une adresse de réponse (Reply–To) différente de l'adresse de l'émetteur (From) :

```
mail("paul@host.com","retour de vacances","excellent",
     "From: michel@toto.fr\nReply-To: eric@toto.fr");
```

C'est aussi grâce à l'en-tête que vous allez être en mesure de dire au gestionnaire de courriels que le message qu'il a reçu doit être considéré comme une page HTML. Les deux lignes suivantes dans l'en-tête indiquent que le courriel n'est pas du simple texte.

```
MIME-Version: 1.0
Content-Type: multipart/alternative; boundary=B97C1230
```

Le corps du courriel doit lui aussi être construit de manière spécifique. Le contenu HTML doit être précédé de :

```
This is a multi-part message in MIME format.
--B97C1230
Content-Type: text/html; charset="iso-8859-1"
```

… et suivi de :

```
--B97C1230--
end of the multi-part
```

La valeur "B97C1230", que l'on retrouve en trois endroits, est une valeur à la fois aléatoire et unique. Il est possible de calculer une valeur unique en PHP de la façon suivante :

```
$val_unique = md5(uniqid(rand()));
```

Affichez une liste de 20 valeurs uniques générées avec cette technique :

```
for ($i=1;$i<=20;$i++)
{
  $unique = md5(uniqid(rand()));
  print("$i - $unique<br>");
}
```

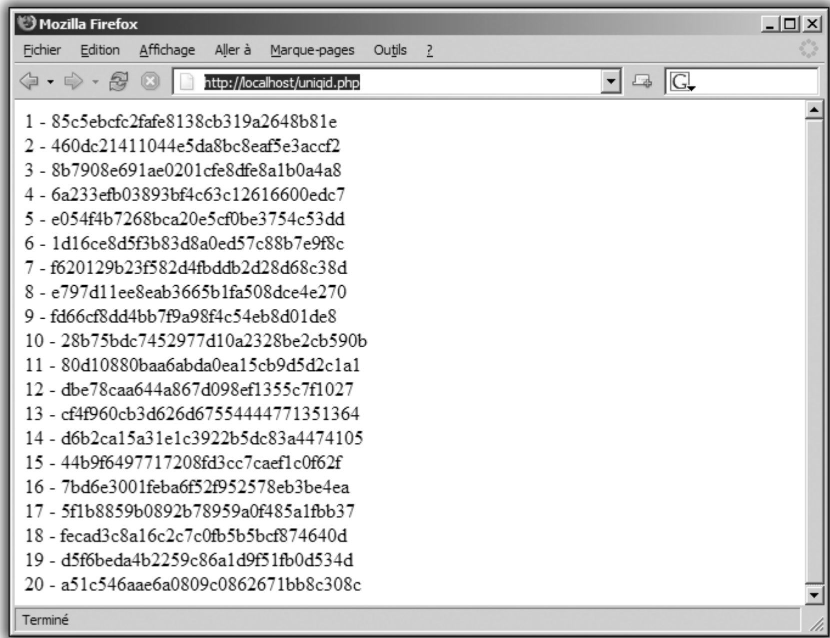


Figure 9.4 : *Liste de 20 valeurs uniques*

Envoyez votre premier courriel en HTML :

```
<?php

$boundary = md5(uniqid(rand()));

$header = "";
$header .= "From: php <test@mondomaine.com>\n";
$header .= "Reply-To: reply@mondomaine.com\n";
$header .= "MIME-Version: 1.0\n";
$header .= "Content-Type: multipart/alternative;
✂ boundary=$boundary\n";

$sujet = "test d’envoi HTML";

$html = "\nThis is a multi-part message in MIME format.";
$html .= "\n--$boundary\nContent-Type: text/html;
✂ charset=\"iso-8859-1\"\n\n";
$html .= "<html><body>\n";
$html .= "<br><br><center><h2><font color=’red’>premier
✂ courriel HTML</font></h2>\n";
$html .= "</body></html>\n";
$html .= "\n--$boundary--\n end of the multi-part";
```

```
mail("test@kernix.com",$sujet,$html,$header);

print("courriel envoyé ...");

?>
```

Le résultat correspond tout à fait à vos attentes :

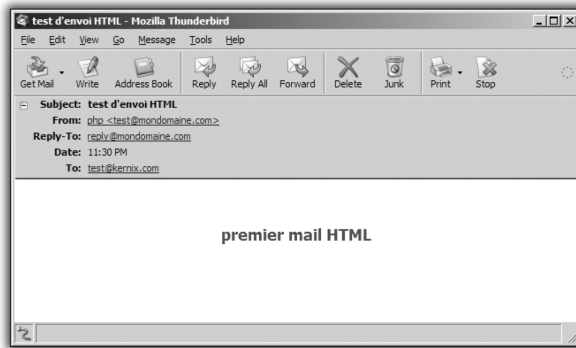


Figure 9.5 : *Un premier courriel HTML*

Pour voir comment la fonction a organisé les données, vous pouvez afficher les sources du courriel :

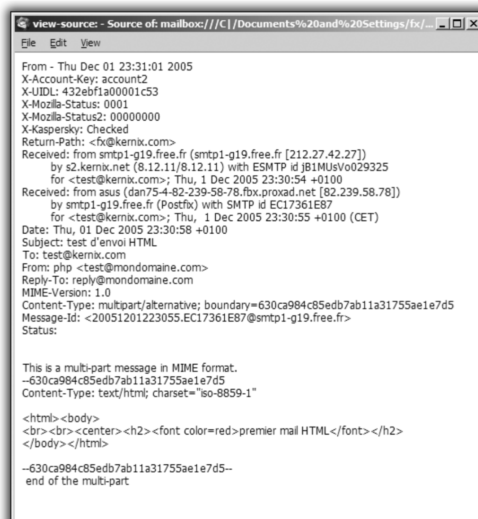


Figure 9.6 : *Source du courriel HTML*

Vous retrouvez bien tous les éléments transmis.



Images et courriels

Il est possible de joindre les images composant le courriel dans le contenu même de ce dernier. Cette méthode complexe n'est cependant pas obligatoire. Il est préférable de placer les images sur un serveur d'hébergement accessible sur le Net et d'y faire appel en utilisant le chemin absolu, par exemple :
`<img src="http://server.com/img/titre.gif" />`.

Reprenez votre exemple d'envoi de profil en ajoutant la dimension HTML :

Listing 9-3 : Envoi des informations dans un courriel HTML

```
$destinataire = "test@kernix.com";

$titre = "réponse au formulaire";

$boundary = md5(uniqid(rand()));

$header = "";
$header .= "From: script php <test@mondomaine.com>\n";
$header .= "Reply-To: reply@mondomaine.com \n";
$header .= "MIME-Version: 1.0\n";
$header .= "Content-Type: multipart/alternative;
✂ boundary=$boundary\n";


$message = "";
$message .= "\nThis is a multi-part message in MIME format.";
$message .= "\n--$boundary\nContent-Type: text/html;
✂ charset=\"iso-8859-1\"\n\n";
$message .= "<html><body>\n";
$message .= "<b>Titre</b> : <font color='red'>".$_REQUEST
✂ ['titre']."</font><br/>\n";
$message .= "<b>Année</b> : ".$_REQUEST['annee']."<br/>\n";
$str_genre = join(',',$_REQUEST['genre']);
$message .= "<b>Genre</b> : ".$str_genre."<br/>\n";
$message .= "<b>Description</b> : ".$_REQUEST
✂ ['description']."<br/>\n";
$message .= "<b>Couleur</b> : ".$_REQUEST['couleur']
✂ ."<br/>\n";
$message .= "<b>Pays</b> : ".$_REQUEST['pays']."<br/>\n";
$str_soustitre = join(',',$_REQUEST['soustitre']);
$message .= "<b>Sous-titres</b> : <i>".$str_soustitre
✂ ."</i><br/>\n";
$message .= "</body></html>\n";
$message .= "\n--$boundary--\n end of the multi-part";
```

```
if (mail($destinataire,$titre,$message,$header)==true) {
  print("<hr/>Les informations ont bien été transmises.");
}
else {
  die("<hr/>L’envoi du courriel a échoué.");
}
```

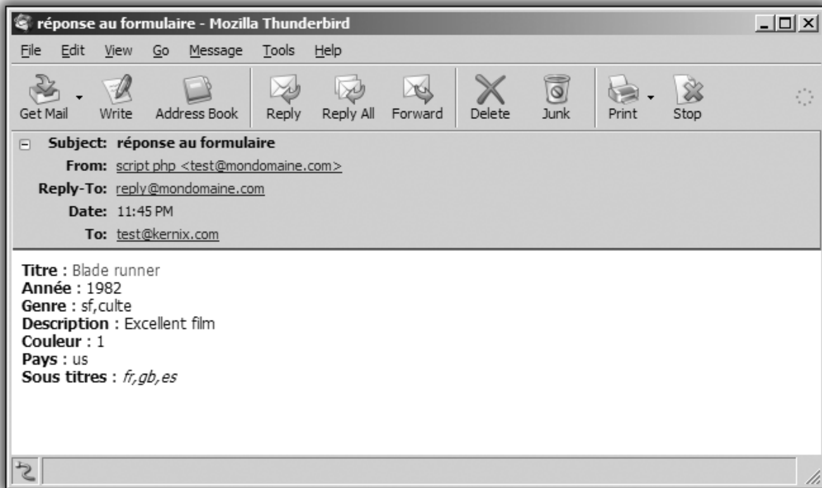


Figure 9.7 : *Courriel en HTML*

Tous les champs sont maintenant bien renseignés (voir le champ `From`).



Le cinquième paramètre

Les versions les plus récentes de PHP (supérieures à 4.0.5) ajoutent un cinquième paramètre optionnel à la fonction `mail()`. Ce paramètre permet de transmettre des « instructions » au logiciel qui va se charger d’envoyer le courriel sur le Net. Parmi les commandes intéressantes, on trouve essentiellement `"-f$adr"`, où `$adr` correspond à l’e-mail de la personne qui recevra un message d’erreur si le courriel ne peut arriver à destination. La fonction `mail()` s’utilise alors comme suit : `mail($destinataire,$titre,$message,$header,"-f$adr");`. Ce paramètre permet aussi d’éviter certaines erreurs avec les serveurs SMTP qui refusent les courriels dont l’origine n’est pas précisée. Ce paramètre n’est pas disponible si PHP est configuré en `safe_mode`.

9.4. Check-list

- L’envoi de courriels en PHP est une opération très simple.
- Les courriels contenant de la couleur ou des images nécessitent la modification de l’en-tête du courriel.
- L’en-tête du courriel permet également de préciser son origine (`From`), l’adresse e-mail de retour et une multitude d’autres informations.
- Le cinquième paramètre de la fonction `mail()` est très utile pour récupérer les messages d’erreur.

Chapitre 10

L’enregistrement dans une base de données

Les bases de données 250
PHP et MySQL 259
Envoi de fichier 277
Le couteau suisse du développeur web : phpMyAdmin 283
Check-list 289

L'objectif de ce chapitre est de réaliser un script capable d'enregistrer dans une base de données des informations transmises par un formulaire représentant une « fiche élève ».

Nous nous attacherons donc à définir de manière simple et précise la notion de bases de données, leur rôle et leur fonctionnement. Nous étudierons ensuite la façon de les interroger depuis un script PHP.

Nous nous intéresserons enfin à phpMyAdmin, outil extrêmement populaire sur le Web qui permet de gérer une base de données MySQL.

10.1. Les bases de données

Les bases de données sont aujourd'hui devenues indispensables et incontournables dans l'univers du Web professionnel. Les plus gros sites mondiaux font tous appel à de tels systèmes.

Qu'est ce qu'un SGBD ?

Pendant longtemps, les développeurs qui créaient des scripts pour le Web (des CGI) utilisaient de simples fichiers texte pour stocker leurs données. Les données étaient enregistrées ligne par ligne, et chaque développeur définissait sa propre norme.

Avec la démocratisation de l'usage des bases de données, il est plus que conseillé d'abandonner aujourd'hui cette approche.

Les avantages offerts par les bases de données sont en effet très nombreux :

- rapidité à tous les niveaux (accès rapide aux données sur le disque) ;
- performances ne diminuant presque pas quand la quantité de données augmente ;
- extractions aisées et pouvant se faire sur des critères complexes ;
- de nombreux outils permettant la sauvegarde, la réplication de données.

Un système de gestion de bases de données (généralement appelé SGBD) est en fait un logiciel dont le seul but est de stocker et de

restituer de l'information le plus rapidement possible. Il s'agit d'un logiciel fonctionnant, le plus souvent, en mode client-serveur, à la manière d'un serveur web ou FTP. Le client envoie des requêtes au serveur et celui-ci retourne une réponse.

Les requêtes sont de types :

```
ENREGISTRE À L'ADRESSE 12 LE NOM Darras et LE PRENOM
Jacques
```

```
RENVOIE TOUTES LES PERSONNES AYANT COMME PRENOM Michel
```

… ou :

```
EFFACE TOUS LES ELEVES AYANT MOINS DE 10 DE MOYENNE
```

Les SGBD sont très nombreux. On trouve parmi les plus connus :

- SQL Server (Microsoft) ;
- PostGRE SQL ;
- Oracle ;
- DB2 ;
- MySQL ;
- SQLite.



Les SGBD en mode fichier

Paradox, dBase, Foxpro, Access, dont vous avez certainement entendu parler, sont des SGBD ne fonctionnant pas en mode client-serveur, mais en mode fichier. Il faut, dans ce cas, disposer à la fois du logiciel et de la base sur son disque dur pour pouvoir les utiliser. Avec l'avènement des réseaux d'entreprise et des applications en ligne, ce mode de fonctionnement risque fort de laisser la place aux SGBD client-serveur.

Dans le cadre de ce livre, vous allez travailler avec MySQL, le SGBD le plus répandu sur le Web, qui possède un certain nombre d'avantages :

- Il est présent chez de nombreux hébergeurs.
- Il est extrêmement rapide (peut-être le plus rapide !).
- Il fonctionne sous Windows, Linux, Mac OS X.
- Il est simple à installer et à utiliser.

- Il est gratuit et open source.
- Il propose une gestion avancée des privilèges et des droits.
- Il existe une compatibilité avec de nombreux langages comme PHP, C, C++, Perl, Python, Java.
- La scalabilité, la sécurité et la stabilité sont irréprochables.
- Il propose des outils tels que phpMyAdmin.
- La gestion de l’encodage des caractères permet de travailler aussi bien avec du texte chinois que français.
- La compatibilité avec le standard SQL est exemplaire.
- La version 5 propose des fonctionnalités avancées, notamment les procédures stockées, les *triggers*, les vues, les *commit/rollback*, les clés étrangères.
- Il permet de changer à la volée la structure des tables et les types des colonnes.

Pour être rigoureux, citons tout de même quelques inconvénients :

- Il n’existe pas de client graphique du niveau de ceux de Microsoft SQL Server ou Oracle.
- Le système de réplication peut encore être amélioré.

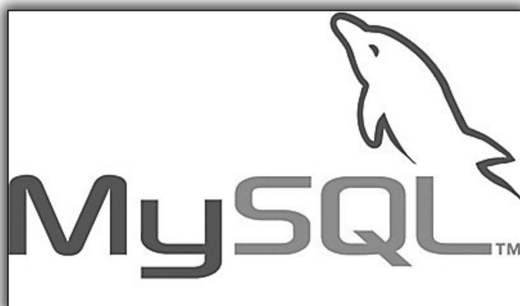


Figure 10.1 : *Le logo de MySQL*

Comme vous l’avez vu, les SGBD ont envahi le Web. On les trouve notamment dans :

- la gestion de contenus (news, forums, blogs) ;
- le commerce électronique, pour la gestion de caddies, de commandes, de profils clients ;
- la gestion des LOG et du trafic ;

- les moteurs de recherche ;
- les plateformes d'enchères.

Ajouter une couche « base de données » à un site web permet généralement de l'enrichir, de le dynamiser et de le professionnaliser. Plus complet, et mis à jour plus régulièrement, votre site incitera d'autant plus facilement les visiteurs à revenir. Avec des modules de newsletters, de votes, de forums, ces mêmes visiteurs pourront, en plus, s'impliquer dans la vie du site et s'y attacher.

Organisation d'un SGBD

Chez la plupart des hébergeurs qui proposent le support des bases de données, vous obtiendrez, lors de l'ouverture de votre compte, quatre informations indispensables :

- le serveur de base de données, par exemple `bdd.kernix`, ou `190.191.192.193` ;
- le nom de la base de données, par exemple `test` ;
- votre identifiant et votre mot de passe, généralement les mêmes que pour le courriel et le FTP, par exemple `monidentifiant`, `monpassword`.

Ce sont ces informations qui vont vous permettre de vous identifier et de communiquer avec le SGBD.

Avant de réaliser vos premières requêtes, il est nécessaire de comprendre comment les données s'organisent au sein d'un SGBD.

Un SGBD est un système hiérarchique et multi-utilisateur. Chaque utilisateur possède certains droits sur certaines bases. Vous ne disposerez généralement de droits que sur une seule base. Une base peut être assimilée à une grosse armoire de rangement dont vous avez la clé (identifiant et mot de passe). Cette armoire va vous permettre d'y placer des classeurs. Les classeurs correspondent aux tables. Chaque base de données contient donc plusieurs tables vous permettant d'organiser vos données. La base *test* pourra par exemple contenir :

- une table *eleve* ;
- une table *professeur* ;
- une table *fournisseur*.

Chaque table contient des caractéristiques qui lui sont propres : on parle alors de colonnes. Les tables ne contiennent pas obligatoirement des données de même type.

- Table : *eleve*.
- Colonnes : *nom*, *prenom*, *adresse*, *ville*, *code postal*, *pays*, *sexe*, *date de naissance*, *taille*, *email*, *telephone*, *langue vivante*.

Lorsque les tables sont créées, il devient possible d'y enregistrer de nouveaux élèves, professeurs ou fournisseurs.

Les données s'organisent alors dans la table ligne par ligne : on parle d'enregistrements. La table *eleve* contient ainsi une liste d'élèves. Chaque élève disposant de ses caractéristiques :

- Dupont – Paul – 12, rue Cronstadt – Paris – 75015…
- Durand – Michel – 23, bd Voltaire – Boulogne – 92100…

Pour différencier de manière unique ces différents enregistrements, il est courant d'ajouter une colonne initiale aux tables : on parle de clé à propos de cette colonne spéciale.

Le premier champ de la table *eleve* devient donc `ideleve` (identifiant de l'élève).

- 1 – Dupont – Paul – 12, rue Cronstadt – Paris – 75015…
- 2 – Durand – Michel – 23, bd Voltaire – Boulogne – 92100…

Plutôt que de demander la ligne où le nom est Dupont et risquer de tomber sur un autre Dupont, il est maintenant possible de demander la ligne ayant la clé 1.

Les requêtes

Présentation du SQL

Comme vous l'avez vu plus haut, vous utilisez des requêtes pour communiquer avec une base. Ces requêtes sont fondées sur un langage propre aux SGBD : le SQL (Structured Query Language). Ce langage est né vers la fin des années 1970 et a vite été normalisé. Par là même, en étudiant le SQL dans le cadre du SGBD MySQL, vous pourrez passer

sans aucune difficulté à un autre SGBD (par exemple PostGRE SQL, l'autre grand SGBD du Web et de l'open source).

Une requête typique en SQL est de la forme :

```
SELECT nom FROM eleve WHERE ideleve = 5
```

Elle pourrait être traduite en français de la manière suivante : « renvoie le nom de l'élève ayant la clé numéro 5 ».

Une présentation approfondie des fonctionnalités les plus avancées du SQL nécessiterait un volume supplémentaire. Vous vous contenterez donc d'étudier les parties du langage nécessaires à la réalisation de la plupart des petites applications web. Vous verrez notamment l'insertion, la sélection, et la suppression de données.

Types de colonnes

Le langage SQL est un langage typé. Une colonne peut être de type chaîne de caractères, nombre entier, à virgule, date, etc. Bien choisir les types de vos données vous permettra d'obtenir de meilleures performances au niveau de vos requêtes. C'est d'ailleurs le premier conseil qui est donné à une personne souhaitant optimiser son application.

Tableau 10.1 : *Les principaux types de données de MySQL*

Type de données MySQL	Définition
`VARCHAR`	Chaîne de caractères de taille variable
`TINYTEXT`	Texte contenant au maximum 255 caractères
`TEXT`	Texte contenant au maximum 65 535 caractères
`MEDIUMTEXT`	Texte contenant au maximum 16 777 215 caractères
`LONGTEXT`	Texte contenant au maximum 4 294 967 295 caractères
`DATETIME`	Date et heure
`DATE`	Date
`TINYINT`	Nombre entier compris entre -128 et 127

Tableau 10.1 : *Les principaux types de données de MySQL*	
Type de données MySQL	**Définition**
`SMALLINT`	Nombre entier compris entre -32 768 et 32 767
`INT`	Nombre entier compris entre -2 147 483 648 et 2 147 483 647
`BIGINT`	Nombre entier compris entre -9 223 372 036 854 775 808 et 9 223 372 036 854 775 807
`FLOAT`	Nombre à virgule



TEXT et BLOB

Les types `TINYBLOB`, `BLOB`, `MEDIUMBLOB`, `LONGBLOB` existent également. Ils ne diffèrent de leurs cousins `TEXT` que par le fait qu'ils sont sensibles à la casse. Les mots `CouCou` et `coucou` sont donc identiques quand ils sont stockés en `TEXT`, et différents en cas de `BLOB`.

À la différence de PHP, les numériques ou les chaînes de caractères sont eux-mêmes divisés en plusieurs types. Cette subdivision a bien évidemment pour objectif de gagner en place et en rapidité. Quand vous voulez stocker un âge, il vaut mieux préférer un `TINYINT` à un `INT`. Votre table sera plus petite et, quand vous voudrez extraire des données, le résultat sera obtenu plus rapidement.

La création d'une table au sein de votre base de données nécessite aussi une requête. La commande utilisée est cette fois `CREATE TABLE`.

La requête SQL permettant de créer votre table *eleve* est la suivante :

```
CREATE TABLE eleve (
  ideleve int(10) unsigned NOT NULL auto_increment,
  nom varchar(64) NOT NULL default '',
  prenom varchar(64) NOT NULL default '',
  adresse varchar(128) NOT NULL default '',
  ville varchar(64) NOT NULL default '',
  cp varchar(8) NOT NULL default '',
  pays varchar(32) NOT NULL default 'france',
  sexe varchar(8) binary NOT NULL default '',
  naissance date NOT NULL default '0000-00-00',
  taille tinyint(10) unsigned NOT NULL default '0',
  email varchar(64) NOT NULL default '',
```

```
  telephone varchar(16) NOT NULL default '',
  lv varchar(16) NOT NULL default '',
  PRIMARY KEY  (ideleve),
  KEY nom (nom)
) TYPE=MyISAM;
```

La table contient 13 colonnes.

Étudions comment est construite cette requête…

Création de la table

```
CREATE TABLE eleve (.
```

Il s'agit de la commande SQL de la requête. Cette dernière fonctionne, en résumé, de la manière suivante :

```
CREATE TABLE nom_de_la_table (Champ1 Type1, Champ2 Type2)
```

Définition du champ ideleve

```
ideleve int(10) unsigned NOT NULL auto_increment,
```

Il s'agit donc d'un entier (`int(10)`) positif (`unsigned`), non nul (`NOT NULL`), qui s'auto-incrémente dès que l'on ajoute une nouvelle entrée. La syntaxe est du type :

```
nom_du_champ type _du_champ liste_d_options
```

Le champ `taille` est aussi un entier positif non nul. En revanche, il ne s'incrémente pas, et possède une valeur par défaut : `0` (`default '0'`). Vous remarquez que nous avons choisi un `TINYINT` pour la taille. Or, la taille peut facilement être supérieure à 127 cm. Associé à l'option `unsigned`, un `TINYINT` peut en fait contenir une valeur allant jusqu'à 255. Il en est de même pour les autres types numériques (le maximum peut être doublé).



Dénomination des tables et des champs

Il est souvent déconseillé d'utiliser des caractères spéciaux dans les programmes pour définir des fonctions ou des variables. De la même manière, en base de données, il est plus sûr d'éviter de mettre des espaces ou des accents dans le nom des tables ou des colonnes. C'est pour cette raison que nous préférons *ideleve* à *idélève*.

Définition du champ nom

```
nom varchar(64) NOT NULL default '',
```

Il s'agit cette fois d'une chaîne de caractères de taille maximale 64, n'ayant pas de valeur par défaut (à la différence du champ `pays` qui a pour valeur par défaut `france`). Plusieurs champs ont le même type sans avoir la même taille. L'idée est d'optimiser vos tables et de ne pas les « alourdir » plus qu'elles n'en ont besoin. Quand on sait que le code postal ne sera jamais plus long que 8 caractères, il est préférable de lui donner une taille de 8. Il faut donc bien calculer, et ne pas voir trop juste. En effet, une donnée comportant 10 caractères, stockée dans un champ de taille 8, sera tronquée.

Le champ `sexe` dispose d'un attribut en plus : `binary`. Cela signifie que le champ en question est sensible à la casse (minuscules, majuscules). Par défaut, ce n'est pas le cas avec de simples `varchar` (par exemple, le champ `ville`). Ainsi, quand on cherche les élèves habitant à Paris, MySQL n'est pas sensible au fait que l'orthographe soit `"paris"`, `"PARIS"` ou `"Paris"`. Ce comportement est souvent celui qu'on attend, mais est plus lourd à générer. Préciser `binary` permet donc également d'accélérer les requêtes.

Définition du champ naissance de type date

```
naissance date NOT NULL default '0000-00-00',
```

Le type `date` a été préféré au type `datetime` car vous n'avez pas besoin de l'heure exacte de la naissance.

Définitions des clés et des index

```
PRIMARY KEY (ideleve), KEY nom (nom)
```

Ces deux lignes permettent de définir les clés et les index sur certaines colonnes : `ideleve` devient une clé primaire, et un index est créé sur la colonne *nom*. Une clé primaire signifie que pour la table *eleve*, il ne peut y avoir qu'un seul élève ayant l'`ideleve 5`. L'index signifie, quant à lui, qu'une table adjacente à la table *eleve* va être créée afin d'accélérer les requêtes basées sur cette colonne indexée. Dans ce cas, il est pertinent d'indexer la colonne *nom*, car il y a de fortes chances que l'on veuille accéder à la fiche d'un élève en précisant son nom. À l'inverse, indexer la colonne *ville* n'est pas très judicieux, car il est très peu probable que l'on souhaite récupérer des enregistrements à partir de

cette colonne. Il faut noter que la création d'un index accompagne systématiquement la création d'une clé primaire. La colonne *ideleve* est donc indexée.



Les index

Les index doivent être utilisés à bon escient. Mettre un index sur une « mauvaise » colonne pourrait conduire à des réponses plus lentes. Il est généralement conseillé de mettre des index sur des colonnes souvent utilisées, et variées au niveau du contenu. Un index « intelligent » peut en revanche permettre d'être 100 fois plus rapide sur certaines requêtes.

Formats de tables

```
"TYPE=MyISAM"
```

MySQL propose différents formats de tables. MyISAM est aujourd'hui le meilleur parmi les formats stables et répandus. Il a l'avantage de disposer de plus de fonctionnalités et d'être plus rapide que son ancêtre ISAM. Préciser le format de la table n'est cependant pas vraiment nécessaire car MySQL crée généralement une table avec le meilleur format dont elle dispose.

10.2. PHP et MySQL

Premières requêtes

Parmi les nombreuses extensions proposées par PHP, MySQL est une des plus utilisées. Les fonctions proposées par cette extension vont permettre de se connecter au serveur de SGBD et à la base. Comme vous l'avez vu plus haut dans ce chapitre, les informations vous permettant de vous y connecter (nom du serveur, nom de la base, identifiant, mot de passe) doivent vous être fournies par votre hébergeur. Si vous travaillez sur votre propre machine, les informations suivantes (qui correspondent à celles par défaut) devraient convenir…

- Serveur SGBD : `localhost`.
- Nom de la base : `test`.
- Identifiant : `root`.
- Mot de passe : vide.

Avant d'envoyer des requêtes, il convient tout d'abord de se connecter au serveur :

```
$liendb = mysql_connect("localhost", "root", "");
```

Choisissez ensuite la base de données avec laquelle vous allez travailler (généralement, vous n'avez pas le choix, vous ne disposez que d'une base de données) :

```
mysql_select_db("test");
```

Il faut ensuite formater une requête en SQL :

```
$sql = "CREATE TABLE eleve (
  ideleve int(10) unsigned NOT NULL auto_increment,
  nom varchar(64) NOT NULL default '',
  prenom varchar(64) NOT NULL default '',
  adresse varchar(128) NOT NULL default '',
  ville varchar(64) NOT NULL default '',
  cp varchar(8) NOT NULL default '',
  pays varchar(32) NOT NULL default 'france',
  sexe varchar(8) binary NOT NULL default '',
  naissance date NOT NULL default '0000-00-00',
  taille int(10) unsigned NOT NULL default '0',
  email varchar(64) NOT NULL default '',
  telephone varchar(16) NOT NULL default '',
  lv varchar(16) NOT NULL default '',
  PRIMARY KEY  (ideleve),
  KEY nom (nom)
)";
```

Envoyez-la ensuite à la base :

```
mysql_query($sql);
```

Lorsque vous n'avez plus besoin de la base de données dans le script, il est conseillé de fermer la connexion au serveur :

```
mysql_close($liendb);
```

Vous comprenez donc que le client dans la relation client/serveur est votre script PHP. C'est lui qui envoie et reçoit les données. Les clients SQL peuvent cependant prendre d'autres formes…

- Une application graphique comme ACCESS (via un lien ODBC) : la création des tables, les requêtes, les liaisons, etc., sont alors réalisées visuellement (voir Figure 8.2).
- Un terminal texte : les requêtes sont ainsi tapées directement en SQL dans un interpréteur de commandes (un shell) et les réponses apparaissent au format texte (voir Figure 8.3).

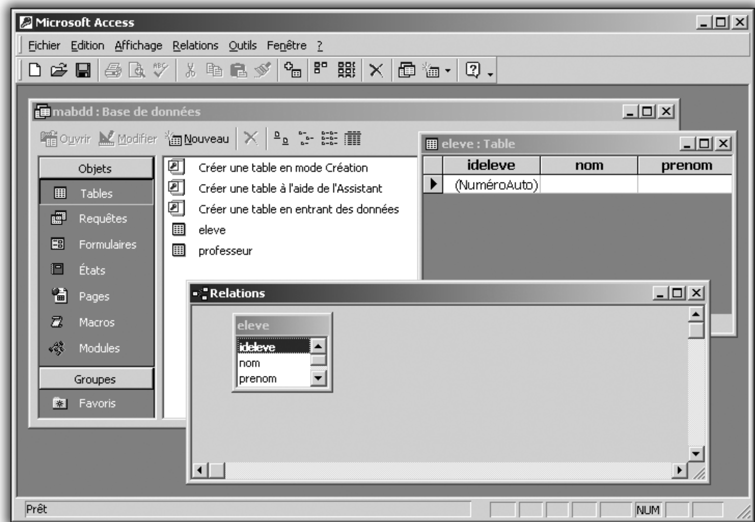


Figure 10.2 : Client SQL graphique

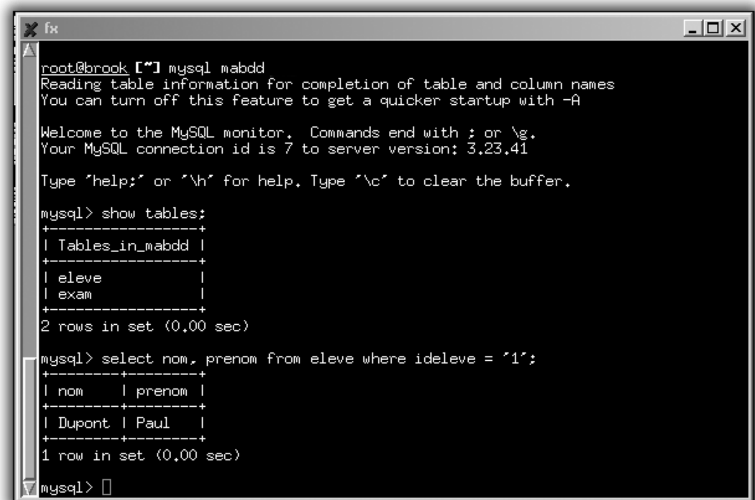


Figure 10.3 : Client SQL texte

Écrivez maintenant le script *creer_table_eleve.php*, qui va vous permettre de créer votre table *eleve* :

Listing 10-1 : Script permettant de créer une table

```
<?php

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db("test");
$sql = "CREATE TABLE eleve (
  ideleve int(10) unsigned NOT NULL auto_increment,
  nom varchar(64) NOT NULL default '',
  prenom varchar(64) NOT NULL default '',
  adresse varchar(128) NOT NULL default '',
  ville varchar(64) NOT NULL default '',
  cp varchar(8) NOT NULL default '',
  pays varchar(32) NOT NULL default 'france',
  sexe varchar(8) binary NOT NULL default '',
  naissance date NOT NULL default '0000-00-00',
  taille int(10) unsigned NOT NULL default '0',
  email varchar(64) NOT NULL default '',
  telephone varchar(16) NOT NULL default '',
  lv varchar(16) NOT NULL default '',
  PRIMARY KEY  (ideleve),
  KEY nom (nom)
)";
mysql_query($sql);
mysql_close($liendb);

echo "table < eleve > créée";

?>
```

Pour résumer, l'exécution de la requête a nécessité cinq grandes étapes…

- Étape 1 : création d'une connexion au serveur.
- Étape 2 : sélection de votre base.
- Étape 3 : préparation de la requête.
- Étape 4 : envoi de la requête à la base.
- Étape 5 : fermeture de votre connexion.

Une fois que vous avez exécuté votre script en appelant `http://localhost/creer_table_eleve.php`, la table *eleve* est alors créée dans votre base *test*.



Fonction mysql_connect() ou mysql_pconnect()

Il existe deux fonctions permettant de se connecter à une base : `mysql_connect()` et `mysql_pconnect()`. La fonction `mysql_pconnect()` utilise une connexion persistante dans le sens où



> elle essaie de trouver une connexion qui a déjà été ouverte avec la base. En pratique, cette fonction n'est à préférer à `mysql_connect()` que lorsque votre site est le seul à fonctionner sur le serveur. Si votre site est hébergé sur un serveur mutualisé (comme c'est le cas la plupart du temps), cette fonction est plutôt à éviter.

Il convient maintenant d'alimenter cette table avec des données. Considérez l'élève suivant…

- Nom : Dupont.
- Prénom : Paul.
- Adresse : 12, rue Brancion.
- Ville : Paris.
- Code postal : 75015.
- Pays : France.
- Sexe : masculin.
- Date de naissance : 11/04/1989.
- Taille : 120 cm.
- Courriel : pdupont@wanadoo.fr.
- Téléphone : 0123456.
- Langue vivante : anglais.

Pour enregistrer cet élève dans la base, vous avez besoin de la commande SQL `INSERT INTO`, dont la syntaxe est la suivante :

```
INSERT INTO nom_table (liste de colonnes) VALUES (liste
des valeurs des colonnes)
```

La requête SQL permettant d'enregistrer Paul Dupont dans la table *eleve* est la suivante :

```
INSERT INTO eleve (nom, prenom, adresse, ville, cp, pays,
sexe, naissance, taille, email, telephone, lv) VALUES
('Dupont', 'Paul', '12 rue Brancion', 'Paris', '75015',
'france', 'masculin', '1989-04-11', '120',
'pdupont@wanadoo.fr', '0123456', 'anglais')
```

Les guillemets ne doivent pas être oubliés autour des données.

La notion d’échappement de caractères existe aussi en SQL, et se gère de la même manière qu’en PHP. Si l’adresse de Paul Dupont avait été « 12, rue de l’Inde », vous auriez dû écrire :

```
INSERT INTO eleve (nom, prenom, adresse, ville, cp, pays,
sexe, naissance, taille, email, telephone, lv) VALUES
('Dupont', 'Paul', '12 rue de l\'Inde', 'Paris',
'75015', 'france', 'masculin', '1989-04-11', '120',
'pdupont@wanadoo.fr', '0123456', 'anglais')
```



Les dates en SQL

Les dates doivent être enregistrées à l’anglo-saxonne : année-mois-jour. Pour une date avec l’heure : `AAAA-MM-JJ HH-MM-SS`.

Écrivez le script *enregistre_pauldupont.php* :

Listing 10-2 : Script enregistre_pauldupont.php

```
<?php

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");
$sql = "INSERT INTO eleve (nom, prenom, adresse, ville,
cp, pays, sexe, naissance, taille, email, telephone, lv)
VALUES ('Dupont', 'Paul', '12 rue Brancion', 'Paris',
'75015', 'france', 'masculin', '1989-04-11', '120',
'pdupont@wanadoo.fr', '0123456', 'anglais')";
mysql_query ($sql);
mysql_close($liendb);

echo "eleve < Paul Dupont > enregistré";

?>
```

En exécutant le script, vous ajoutez votre premier enregistrement à la table. L’`ideleve` de cet enregistrement est `1` car il s’agit d’une colonne qui s’auto-incrémente à chaque insertion.

Notez que la requête SQL en elle-même n’est qu’une chaîne de caractères toute simple. Vous auriez aussi pu écrire :

```
$command = "INSERT INTO";
$table = "eleve";
$colonnes = "nom, prenom, adresse, ville, cp, pays, sexe,
            naissance, taille, email, telephone, lv";
$valeurs = "'Dupont', 'Paul', '12 rue Brancion', 'Paris',
          '75015', 'france', 'masculin', '1989-04-11', '120',
          'pdupont@wanadoo.fr', '0123456', 'anglais'";
```

```
$sql = "$command $table ($colonnes) VALUES ($valeurs)";
```

Vous n'avez fait jusqu'alors qu'envoyer des requêtes à la base et vous n'avez pas eu besoin de récupérer des données. Vous allez donc maintenant écrire un script qui va vous renvoyer le nom et le prénom de l'élève qui a la clé `1`. Après `CREATE TABLE` et `INSERT INTO`, la commande SQL dont vous allez avoir besoin est `SELECT` :

```
SELECT liste des colonnes sélectionnées FROM nom de la
table WHERE clause
```

Dans cet exemple, vous voulez récupérer les données présentes dans les colonnes *nom* et *prenom* de l'enregistrement ayant pour `ideleve` la valeur `1`. Cela se traduit ainsi :

```
SELECT nom, prenom FROM eleve WHERE ideleve = '1'
```

Écrivez le script *voir_pauldupont.php* :

```
<?php

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");
$sql = "SELECT nom, prenom FROM eleve WHERE ideleve =
✂ '1'";
$resultat = mysql_query ($sql);
$eleve = mysql_fetch_array ($resultat);
$nom = $eleve['nom'];
$prenom = $eleve['prenom'];
echo "eleve [1], nom = $nom, prenom = $prenom";
mysql_close($liendb);

?>
```

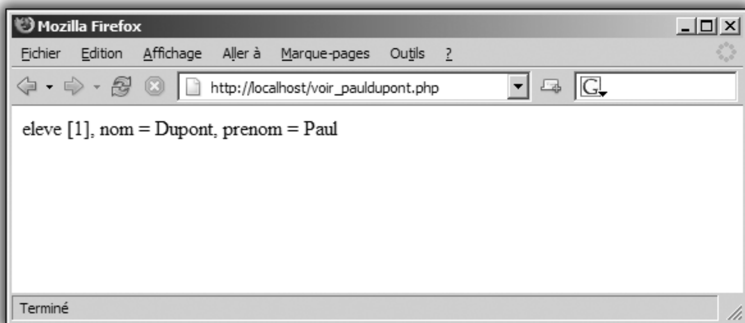


Figure 10.4 : *Résultat sans surprise de votre premier SELECT*

Dans ce script, et à la différence des précédents, vous récupérez le résultat de la requête dans une variable `$resultat`. Cette variable pointe sur l'ensemble des données renvoyées par le serveur de BDD : on la qualifie d'identifiant de résultat.

Pour accéder à ces données, vous utilisez la fonction `mysql_fetch_array()`, qui crée le tableau `$eleve[]` et qui permet d'accéder aux différentes colonnes contenues dans la réponse : `$eleve['nom']` retourne ainsi le nom de l'élève. Il ne servirait à rien d'écrire `$eleve['ville']`, car vous n'avez pas demandé au serveur la colonne *ville*.

Dans cet exemple, vous ne récupérez qu'un enregistrement. Dans la majorité des cas, cependant, les requêtes faites aux SGBD renvoient plusieurs enregistrements. C'est d'ailleurs précisément la puissance des bases de données que de pouvoir extraire des données suivant des critères très précis.

Avant de pouvoir extraire plusieurs enregistrements, il faut remplir votre table :

Listing 10-3 : Trois nouveaux élèves enregistrés

```
<?php

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");
$sql = "INSERT INTO eleve (nom, prenom, adresse, ville, cp,
      pays, sexe, naissance, taille, email, telephone, lv)
      VALUES ('Pitel', 'Guillaume', '5 rue des Sorcières',
              'Paris', '75013', 'france', 'masculin',
              '1989-05-21', '130', 'pitel@limsi.fr',
              '0456789', 'allemand')";
mysql_query ($sql);
$sql = "INSERT INTO eleve (nom, prenom, adresse, ville,
       cp, pays, sexe, naissance, taille, email, telephone, lv)
       VALUES ('Metayer', 'Fabrice', '123 bd de Bretagne',
               'Paris', '75015', 'france', 'masculin',
               '1989-11-3', '95', 'fm@hotmail.com',
               '0789456', 'anglais')";
mysql_query ($sql);
$sql = "INSERT INTO eleve (nom, prenom, adresse, ville, cp,
       pays, sexe, naissance, taille, email, telephone, lv)
       VALUES ('Marillier', 'Olivia', '32 avenue du Golf',
               'Paris', '75015', 'france', 'feminin',
               '1988-10-19', '145', 'olivia@marillier.fr',
               '0741852', 'espagnol')";
```

```
mysql_query ($sql);
mysql_close($liendb);

echo "3 élèves enregistrés";

?>
```



Plusieurs requêtes

Dans ce script, vous envoyez plusieurs requêtes à la base. La connexion (`mysql_connect()`) et la déconnexion (`mysql_close()`) n'ont cependant lieu qu'une fois, au début et à la fin du script.

Vérifiez que le nombre d'enregistrements dans la table est bien de quatre :

```
<?php

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");
$sql = "SELECT ideleve FROM eleve";
$resultat = mysql_query ($sql);
$nb_eleves = mysql_num_rows($resultat);
echo "< $nb_eleves > élèves dans la table eleve";
mysql_close($liendb);

?>
```

La fonction permettant d'obtenir le nombre d'enregistrements est `mysql_num_rows()`. Elle prend en argument le pointeur sur les données renvoyées par MySQL : `$resultat` (rappelons que *row* signifie « ligne »).

Dans ce script, vous n'ajoutez pas `WHERE` à la fin du `SELECT` car vous voulez récupérer tous les élèves.

Vous avez vu, en revanche, avec le script *voir_pauldupont.php*, qu'il est possible d'extraire des données de la base selon certaines contraintes. Ces contraintes se placent derrière `WHERE` et sont réalisées sur le nom des colonnes. Si vous souhaitez, par exemple, avoir la liste des élèves de taille supérieure ou égale à 150 cm, vous utilisez la requête suivante :

```
SELECT * FROM eleve WHERE taille >= '150'
```

L'astérisque (`*`) indique que vous souhaitez récupérer toutes les informations (colonnes) des élèves dont la taille est inférieure à 150 cm. Si vous aviez voulu ne récupérer que leur taille, vous auriez dû écrire :

```
SELECT taille FROM eleve WHERE taille <= '150'
```

Il est possible de combiner plusieurs contraintes avec `AND` :

```
SELECT nom FROM eleve
WHERE ville = 'paris' AND sexe = 'masculin'
```

Cette requête retourne le nom des garçons parisiens. Pour sélectionner les élèves étudiant l’espagnol ou l’anglais, la liaison entre les contraintes devient `OR` :

```
SELECT * FROM eleve
WHERE lv = 'anglais' OR prenom = 'espagnol'
```

Les extractions par rapport aux dates se calquent sur le même modèle. Listez, par exemple, les élèves nés après 1989 et n’habitant pas dans le XVe arrondissement :

```
SELECT prenom FROM eleve
WHERE naissance > '1989-01-01' AND cp != '75015'
```

Écrivez maintenant le script complet qui permet d’afficher les noms et les prénoms des garçons parisiens :

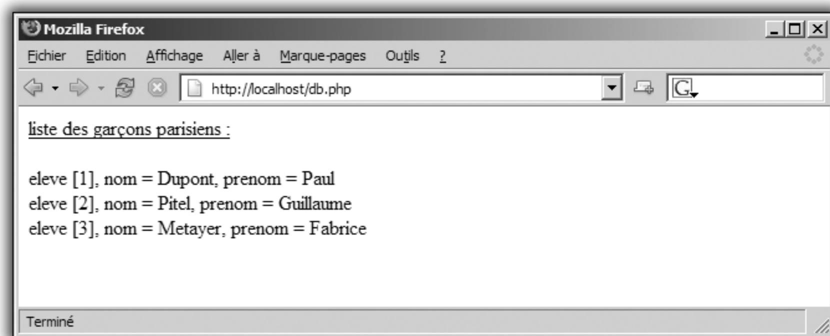


Figure 10.5 : *Liste des garçons parisiens*

```
<?php

echo "<u>liste des garçons parisiens :</u> <br><br>";

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");
$sql = "SELECT * FROM eleve WHERE ville = 'paris'
AND sexe = 'masculin'";
$resultat = mysql_query ($sql);
while ($eleve = mysql_fetch_array ($resultat))
{
```

```
  $id = $eleve['ideleve'];
  $nom = $eleve['nom'];
  $prenom = $eleve['prenom'];
  echo "eleve [$id], nom = $nom, prenom = $prenom<br>";
}
mysql_close($liendb);

?>
```

L'intérêt de ce script se situe au niveau de la boucle `while ($eleve = mysql_fetch_array ($resultat))`. Vous vous apercevez qu'à chaque itération vous passez à l'élève suivant. Vous pouvez envisager `$resultat` comme une variable qui contient à la fois le résultat retourné par le SGBD et un pointeur dirigé sur la ligne en cours. La fonction `mysql_fetch_array()` fait, en réalité, avancer le pointeur à chaque fois qu'elle est appelée. Si elle est appelée et si le pointeur est déjà à la fin du résultat, `mysql_fetch_array()` retourne `false` et la boucle s'arrête.

Vous auriez aussi pu passer par une boucle `for` en vous appuyant sur la fonction `mysql_num_rows()` :

```
<?php

echo "<u>liste des garçons parisiens :</u> <br><br>";

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");
$sql = "SELECT * FROM eleve WHERE ville = 'paris'
AND sexe = 'masculin'";
$resultat = mysql_query ($sql);
$n = mysql_num_rows($resultat);
for ($i = 1; $i <= $n; $i++)
{
  $eleve = mysql_fetch_array ($resultat);
  $id = $eleve['ideleve'];
  $nom = $eleve['nom'];
  $prenom = $eleve['prenom'];
  echo "eleve [$id], nom = $nom, prenom = $prenom<br>";
}
mysql_close($liendb);

?>
```

Le résultat est le même, mais vous utilisez une fonction supplémentaire.

Une autre fonction peut être utilisée pour accéder aux différentes lignes et colonnes : `mysql_result()`. Cette fonction prend en paramètre la variable `$resultat`, le numéro de la ligne et le nom de la colonne :

```
<?php

echo "<u>liste des garçons parisiens :</u> <br><br>";

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");
$sql = "SELECT * FROM eleve WHERE ville = 'paris'
AND sexe = 'masculin'";
$resultat = mysql_query ($sql);
$n = mysql_num_rows($resultat);
for ($i = 0; $i < $n; $i++)
{
  $id = mysql_result($resultat,$i,'ideleve');
  $nom = mysql_result($resultat,$i,'nom');
  $prenom = mysql_result($resultat,$i,'prenom');
  echo "eleve [$id], nom = $nom, prenom = $prenom<br>s";
}
mysql_close($liendb);

?>
```

Il est aussi possible de passer le numéro de la colonne plutôt que son nom. La première colonne a comme indice `0`. Vous allez pouvoir lister tous les champs sans avoir à écrire leur nom :

```
<?php

echo "<u>liste des garçons parisiens :</u> <br><br>";

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");
$sql = "SELECT * FROM eleve WHERE ville = 'paris'
AND sexe = 'masculin'";
$resultat = mysql_query ($sql);
$n = mysql_num_rows($resultat);
for ($i = 0; $i < $n; $i++)
{
  for ($j = 0; $j < 13; $j++)
     echo mysql_result($resultat,$i,$j) . " ";
  echo "<br>";
}
mysql_close($liendb);

?>
```

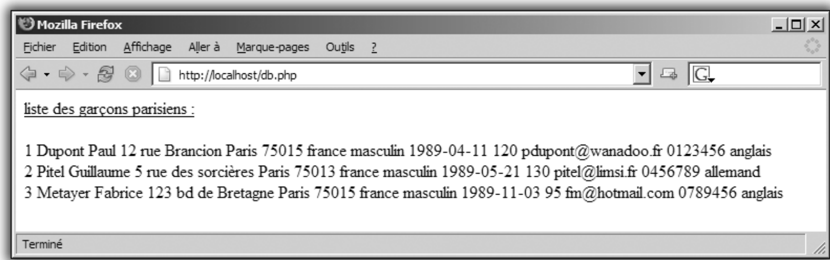


Figure 10.6 : *Liste des garçons parisiens avec tous les champs*

Le script contient deux boucles `for` imbriquées. La première permet de passer d'une ligne à la suivante, la seconde fait passer d'une colonne à l'autre.

Allons plus loin dans ce sens : essayez de réaliser un script qui, quelle que soit la requête, liste toutes les lignes et les colonnes du résultat. Vous allez avoir besoin pour cela de la fonction `mysql_fields()`. Cette commande retourne le nombre de colonnes dans le résultat, ce qui permet d'ignorer la valeur `13` que vous avez écrite « en dur » dans le code :

```
<?php

$liendb = mysql_connect("localhost", "root", "");

mysql_select_db ("test");

$sql = "SELECT nom, prenom, lv FROM eleve
        WHERE ville = 'paris'";
$resultat = mysql_query ($sql);

$nb_lignes = mysql_num_rows($resultat);
$nb_colonnes = mysql_num_fields($resultat);

echo "résultat de la requête : <i>$sql</i> <hr>";
echo "<table border=1 width=100%>";

for ($i = 0; $i < $nb_lignes; $i++)
{
  echo "<tr>";
  for ($j = 0; $j < $nb_colonnes; $j++)
      echo "<td>" . mysql_result($resultat,$i,$j) . "</td>";
  echo "</tr>";
}
echo "</table>";

mysql_close($liendb);

?>
```

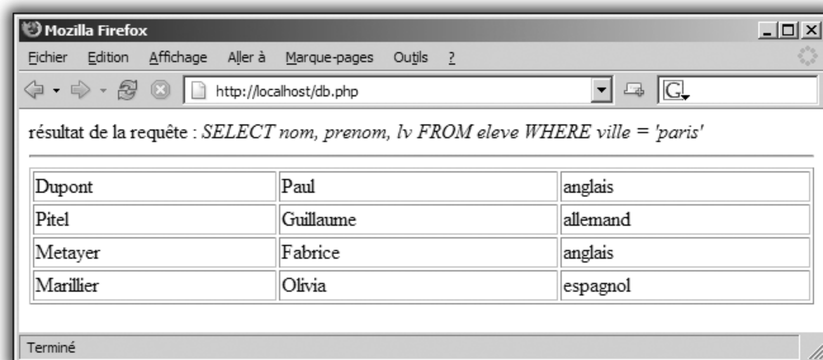


Figure 10.7 : *Affichage générique du résultat d'une requête*

Si vous changez la requête, le tableau est automatiquement mis à jour :

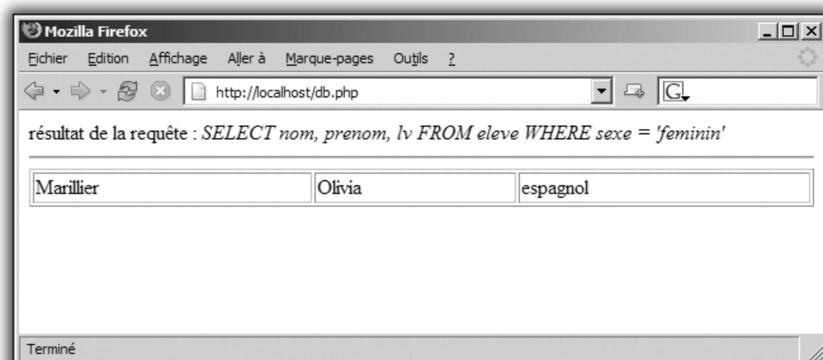


Figure 10.8 : *Le tableau est mis à jour*

Enregistrement d'une fiche

Réalisez maintenant le script *eleve_enregistre.php* qui va permettre d'enregistrer un élève dans la base.

Commencez par créer le formulaire *eleve.html* :

Listing 10-4 : Fichier ajout_eleve.html

```
<html>

<head><title>Ajouter un élève</title></head>

<body>

<h1>Ajouter un élève :</h1>

<form action="eleve_enregistre.php" method="post">

<label>nom</label>
<input type="text" name="nom" /><br/>

<label>prénom</label>
<input type="text" name="prenom" /><br/>

<label>adresse</label>
<textarea name="adresse"></textarea><br/>

<label>ville</label>
<input type="text" name="ville" /><br/>

<label>code postal</label>
<input type="text" name="codepostal" /><br/>

<label>pays</label>
<input type="text" name="pays" /><br/>

<label>sexe</label>
<input type="radio" name="sexe" value="masculin" /> M -
<input type="radio" name="sexe" value="feminin" /> F<br/>

<label>date naissance</label>
<input type="text" name="naissance" /><br/>

<label>taille (cm)</label>
<input type="text" name="taille" /><br/>

<label>email</label>
<input type="text" name="email" /><br/>

<label>téléphone</label>
<input type="text" name="telephone" /><br/>

<label>langue vivante</label>
<select name="lv">
 <option value="anglais">anglais</option>
 <option value="espagnol">espagnol</option>
 <option value="allemand">allemand</option>
```

```
</select>

<br/>
<br/>
<input type="submit" value="enregistrer" />

</form>

</body>
</html>
```

Figure 10.9 : *Formulaire d’ajout d’élève*

Vos scripts, pour le moment, récupèrent des données d’un formulaire ou envoient des requêtes à MySQL. L’idée est maintenant de combiner les deux en composant la requête SQL à partir des données transmises. Les variables `$_REQUEST['nom']` ont remplacé les valeurs fixes que vous aviez l’habitude de mettre (`Dupont`).

Listing 10-5 : Script eleve_enregistre.php

```
<?php

if (empty($_REQUEST['nom']) || empty($_REQUEST['prenom']) ||
    empty($_REQUEST['adresse']) || empty($_REQUEST['ville']) ||
    empty($_REQUEST['codepostal']) || empty($_REQUEST['pays']) ||
```

```
    empty($_REQUEST['naissance']) || empty($_REQUEST['telephone']) ||
    empty($_REQUEST['lv']))
    die("ERREUR : tous les champs doivent être remplis.");

if ($_REQUEST['sexe']!="masculin" &&
    $_REQUEST['sexe']!="feminin")
    die("ERREUR : choisissez votre sexe.");

if (preg_match("/^[\w\.-]+@[\w\.-]+\.[a-z]{2,3}$/i",
               $_REQUEST['email']) == false)
    die("ERREUR : adresse e-mail non valide.");

if ($_REQUEST['taille']<=100 || $_REQUEST['taille']>=220)
    die("ERREUR : la taille n'est pas valide.");

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");
$sql = "INSERT INTO eleve (nom, prenom, adresse, ville,
cp, pays, sexe, naissance, taille, email, telephone, lv)
VALUES ('".$_REQUEST['nom']."', '".$_REQUEST['prenom']."', '"
.$_REQUEST['adresse']."', '".$_REQUEST['ville']."', '"
.$_REQUEST['codepostal']."', '".$_REQUEST['pays']."', '"
.$_REQUEST['sexe']."', '".$_REQUEST['naissance']."', '"
.$_REQUEST['taille']."', '".$_REQUEST['email']."', '"
.$_REQUEST['telephone']."', '"
.$_REQUEST['lv']."')";

if (mysql_query($sql)!=false)
    print("eleve < ".$_REQUEST['nom']." > enregistré.");
else
    print("Echec lors de la création de la fiche.");

mysql_close($liendb);

?>
```

Vous remarquez qu'un test est fait pour savoir si l'exécution de la requête s'est bien déroulée. La fonction `mysql_query()` retourne en effet `false` en cas d'échec.

Il est souvent intéressant de savoir quel est le numéro de la ligne qui vient d'être enregistré. Si la table contient une colonne fonctionnant en `AUTO_INCREMENT`, la fonction `mysql_insert_id()` permet d'avoir accès à la valeur générée par le dernier `INSERT`.

Modifiez votre script de manière à ce qu'il affiche aussi le numéro de l'`ideleve` dans la réponse :

```
if (mysql_query($sql)!=false) {
  $ideleve = mysql_insert_id();
  print("eleve [$ideleve] < ".$_REQUEST['nom']." >
  ✂ enregistré.");
}
else
  print("Echec lors de la création de la fiche.");
```

En analysant bien ce script (et en connaissant le problème de l'échappement de caractère), vous pourriez vous demander si ce que vous écrivez est valide. En effet, si l'internaute a tapé `"34 rue de l'Isle"` comme adresse dans le formulaire, la requête reste-t-elle correcte malgré la présence des guillemets ? Est-il nécessaire d'échapper, avec une barre oblique inversée, toutes les variables provenant du formulaire ?

Listing 10-6 : Exemple de requête entièrement échappée

```
$sql = "INSERT INTO eleve (nom, prenom, adresse, ville,
cp, pays, sexe, naissance, taille, email, telephone, lv)
VALUES ('" . addslashes($_REQUEST['nom']) ."', '" .
addslashes($_REQUEST['prenom']) ."', '" .
addslashes($_REQUEST['adresse']) ."', '" .
addslashes($_REQUEST['ville']) ."', '" .
addslashes($_REQUEST['codepostal']) ."', '" .
addslashes($_REQUEST['pays']) ."', '" .
addslashes($_REQUEST['sexe']) ."', '" .
addslashes($_REQUEST['naissance']) ."', '" .
addslashes($_REQUEST['taille']) ."', '" .
addslashes($_REQUEST['email']) ."', '" .
addslashes($_REQUEST['telephone']) ."', '" .
addslashes($_REQUEST['lv']) ."')";
```

La réponse est assez complexe, car PHP gère cette problématique d'une manière très spéciale.

Par défaut, en PHP, toutes les variables d'un script provenant d'un formulaire ou d'un cookie sont déjà échappées : on parle de *magic quotes* à propos de cet échappement automatique.

Ainsi, lorsque l'internaute envoie `"34 rue de l'Isle"`, le script reçoit une variable `$_REQUEST['adresse']` contenant `"34 rue de l\'Isle"`. Il est donc indispensable de savoir si votre système fonctionne sur ce modèle car si vous échappez une telle variable, son contenu est alors doublement échappé et contient `"34 rue de l\\\'Isle"`. La fonction PHP qui vous permet de statuer sur la question est `get_magic_quotes_gpc()` :

```
if (get_magic_quotes_gpc())
  echo "le système utilise les 'magic quotes'";
else
  echo "le système n'utilise pas le système des 'magic quotes'";
```

Si ce script affiche `"le système utilise les 'magic quotes'"`, votre requête d'origine est valide ; sinon, c'est la deuxième solution (avec les `addslashes()`) qui doit être utilisée.

L'idéal est donc, avant de commencer à développer un applicatif, de vérifier comment fonctionne l'hébergeur avec qui vous allez travailler. Votre code sera en effet différent selon qu'il autorise ou qu'il n'autorise pas les *magic quotes*. La première alternative reste cependant la plus répandue.



Variables initialisées dans le script

Il faut bien faire attention que les *magic quotes* n'interviennent que sur les variables provenant d'un formulaire ou d'un cookie, et non sur les variables définies (ou redéfinies) au sein même du script. Si vous composez une requête de la façon suivante, vous risquez fort de vous retrouver avec un message d'erreur :

```
$adresse = "34 rue de l'Isle";

$sql = "INSERT INTO test (adresse) VALUES ('$adresse')";
```

Comme la variable est définie au sein du script, la requête doit être écrite de cette manière :

```
$sql = "INSERT INTO test (adresse) VALUES ('"
. addslashes($adresse) . "')";
```

10.3. Envoi de fichier

Vous allez maintenant enrichir votre fiche et ajouter une photo au profil des élèves.

Modification de la structure d'une table

Une colonne *photo* doit donc être insérée dans la table. C'est la commande SQL `ALTER` qui va permettre de modifier la structure de la table :

```
ALTER TABLE eleve ADD photo VARCHAR(64) NULL
```

Cette requête ajoute une colonne nommée *photo*, de type chaîne de caractères, de taille 64, et pouvant être vide (`NULL`). La commande `ALTER` permet ainsi de changer la structure d’une table en ajoutant, effaçant, modifiant des colonnes. Ajoutez une colonne *essai* à la table *eleve* afin de procéder à quelques tests :

```
ALTER TABLE eleve ADD essai INT(10) unsigned NOT NULL
```

La colonne *essai* est donc un entier positif non nul. Faites en sorte maintenant qu’elle devienne un nombre flottant :

```
ALTER TABLE eleve CHANGE essai essai FLOAT
```

Modifiez maintenant son nom, et appelez-la `essaie2` :

```
ALTER TABLE eleve CHANGE essai essai2 FLOAT
```

Finalement, supprimez cette colonne de la table *eleve* :

```
ALTER TABLE eleve DROP essai2
```

Envoi de fichier

L’envoi d’un fichier impose l’utilisation de la méthode `POST` et d’un encodage spécial des données transmises. Le changement d’encodage est rendu possible grâce à l’attribut `enctype` qui prend alors la valeur `"multipart/form-data"`.

Il vous est possible de choisir la photo sur votre disque dur en utilisant un widget spécifique pour le champ `photo` : l’`INPUT` de type `FILE`.

Votre formulaire devient donc :

Listing 10-7 : form.html

```
<html>

<head><title>Ajouter un élève</title></head>

<body>

<h1>Ajouter un élève :</h1>

<form action="eleve_enregistre.php" method="post"
      enctype="multipart/form-data">

<label>nom</label>
<input type="text" name="nom" /><br/>
```

```
<label>prénom</label>
<input type="text" name="prenom" /><br/>

<label>adresse</label>
<textarea name="adresse"></textarea><br/>

<label>ville</label>
<input type="text" name="ville" /><br/>

<label>code postal</label>
<input type="text" name="codepostal" /><br/>

<label>pays</label>
<input type="text" name="pays" /><br/>

<label>sexe</label>
<input type="radio" name="sexe" value="masculin" /> M -
<input type="radio" name="sexe" value="feminin" /> F<br/>

<label>date naissance</label>
<input type="text" name="naissance" /><br/>

<label>taille (cm)</label>
<input type="text" name="taille" /><br/>

<label>email</label>
<input type="text" name="email" /><br/>

<label>téléphone</label>
<input type="text" name="telephone" /><br/>

<label>langue vivante</label>
<select name="lv">
 <option value="anglais">anglais</option>
 <option value="espagnol">espagnol</option>
 <option value="allemand">allemand</option>
</select><br/>

<label>photo</label>
<input type="file" name="photo" /><br/>

<br/>
<br/>
<input type="submit" value="enregistrer" />

</form>

</body>
</html>
```

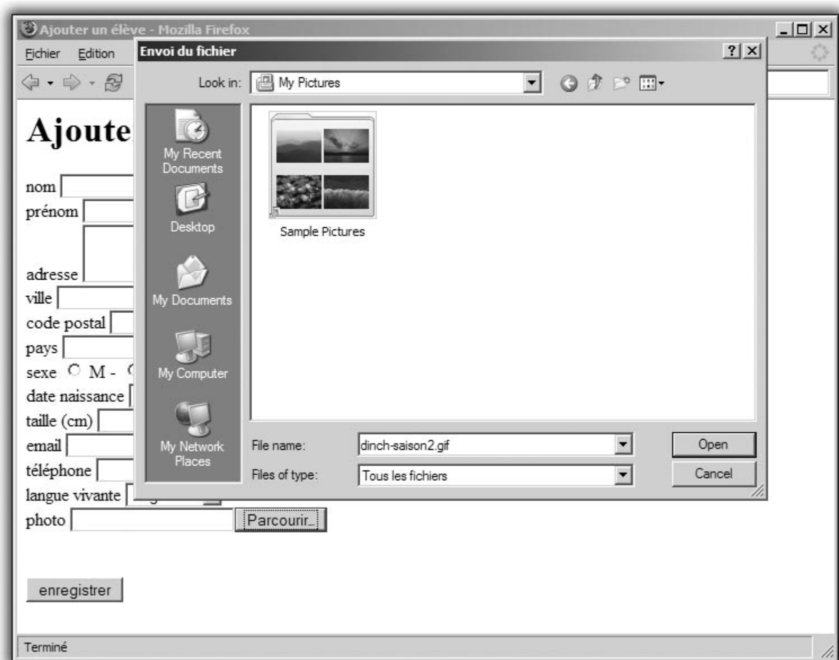


Figure 10.10 : *Il est possible avec le widget file de sélectionner une photo sur son disque dur*

Il convient maintenant de modifier votre script d'enregistrement afin qu'il place la photo dans votre compte, qu'il affiche la photo et qu'il enregistre le profil de l'élève (nom de la photo incluse).

PHP rend l'envoi de fichier très facile. Deux fonctions sont utilisées : `is_uploaded_file()` et `move_uploaded_file()`.

La fonction `is_uploaded_file()` permet de vérifier que l'envoi du fichier s'est déroulé sans encombre. Elle prend en paramètre un nom temporaire de fichier et retourne un booléen en cas de succès (`true`) ou d'échec (`false`). Le nom temporaire se récupère via la super-globale `$_FILES`. Cette variable est un tableau dont les clés correspondent aux noms des widgets de type `file`. Dans le cas présent, `$_FILES['photo']` est créé et propose deux éléments :

- `$_FILES['photo']['tmp_name']` qui contient le nom temporaire du fichier sur le serveur ;
- `$_FILES['photo']['name']` qui contient le nom du fichier que vous avez sélectionné sur votre machine.

La fonction s'utilise donc de la manière suivante:

```
if (is_uploaded_file($_FILES['photo']['tmp_name'])==true)
// on passe à la suite
```



Nom temporaire

Les plus curieux auront peut-être eu la curiosité d'afficher `$_FILES['photo']['tmp_name']` et se seront alors aperçus que le nom temporaire n'a vraiment aucun rapport avec le fichier sélectionné. Ce nom peut être de la forme `'/tmp/phpDVzJ1N'`.

Le déplacement du fichier temporaire est réalisé par la fonction `move_uploaded_file()`. Son premier paramètre correspond au nom du fichier temporaire (`$_FILES['photo']['tmp_name']`) et le second à l'emplacement final du fichier. Cet emplacement contient à la fois le répertoire qui contiendra le fichier et son nom. Généralement, le nom du fichier est celui d'origine : `$_FILES['photo']['name']`. Le répertoire en revanche peut se situer n'importe où. Si vous voulez déplacer le fichier dans un répertoire *images* se trouvant au même niveau que votre script, utilisez le chemin :

```
"images/".$_FILES['photo']['name']
```

Si le répertoire *images* se situe un niveau au-dessus de votre script, le chemin devient :

```
"../images/".$_FILES['photo']['name']
```



Taille du fichier

La taille du fichier transmis se récupère de la manière suivante : `$_FILES['photo']['size']`.

Contentez-vous pour l'instant de placer le fichier au même niveau que le script *eleve_enregistre.php* :

Listing 10-8 : Transmission d'une photo

```
<?php

if (is_uploaded_file($_FILES['photo']['tmp_name'])) {
  move_uploaded_file($_FILES['photo']['tmp_name'],
                     $_FILES['photo']['name']);
  print("<center><img src='".$_FILES['photo']['name'].
        "' /></center></hr>");
}
```

```
else {
  die("Problème d’envoi du fichier.");
}

if (empty($_REQUEST['nom']) || empty($_REQUEST['prenom']) ||
    empty($_REQUEST['adresse']) ||
    empty($_REQUEST['ville']) ||
    empty($_REQUEST['codepostal']) ||
    empty($_REQUEST['pays']) ||
    empty($_REQUEST['naissance']) ||
    empty($_REQUEST['telephone']) ||
    empty($_REQUEST['lv']))
    die("ERREUR : tous les champs doivent être remplis.");

if ($_REQUEST['sexe']!="masculin" &&
    $_REQUEST['sexe']!="feminin")
    die("ERREUR : choisissez votre sexe.");

if (preg_match("/^[\w\.-]+@[\w\.-]+\.[a-z]{2,3}$/i",
    $_REQUEST['email']) == false)
    die("ERREUR : adresse e-mail non valide.");

if ($_REQUEST['taille']<=100 || $_REQUEST['taille']>=220)
    die("ERREUR : la taille n’est pas valide.");

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");
$sql = "INSERT INTO eleve (nom, prenom, adresse, ville,
cp, pays, sexe, naissance, taille, email, telephone, lv,
photo) VALUES ('".$_REQUEST['nom']."', '"
.$_REQUEST['prenom']."', '".$_REQUEST['adresse']."', '"
.$_REQUEST['ville']."', '".$_REQUEST['codepostal']."', '"
.$_REQUEST['pays']."', '".$_REQUEST['sexe']."', '"
.$_REQUEST['naissance']."', '".$_REQUEST['taille']."', '"
.$_REQUEST['email']."', '".$_REQUEST['telephone']."', '"
.$_REQUEST['lv']."', '".$_FILES['photo']['name']."')";

$ideleve = mysql_insert_id();

if (mysql_query($sql)!=false) {
  $ideleve = mysql_insert_id();
  print("eleve [$ideleve] < ".$_REQUEST['nom']." > enregistré.");
}
else
  print("Echec lors de la création de la fiche.");

mysql_close($liendb);

?>
```



Figure 10.11 : *Transmission de la photo*



Écrasement de fichiers

Un fichier portant le même nom qu'un fichier déjà présent sur le serveur remplacera ce dernier en étant uploadé.

Plusieurs fichiers peuvent être envoyés au sein d'un même formulaire. Il faut alors créer plusieurs champs de type `file`, chacun disposant de son propre attribut `name`. Au niveau du script, vous vous retrouvez alors avec autant de variables que de champs. Si vous avez les champs `photo1` et `photo2`, les variables suivantes sont créées : `$_FILES['photo1']` et `$_FILES['photo2']`.



Taille maximale des fichiers transmis

La taille des fichiers uploadés est généralement limitée afin d'éviter les abus. Vous pouvez obtenir cette valeur limitée en utilisant la fonction `phpinfo()` qui affichera la valeur d'`upload_max_filesize`.

10.4. Le couteau suisse du développeur web : phpMyAdmin

L'outil phpMyAdmin est développé en PHP et permet de gérer intégralement votre base.

Les fonctionnalités qu'il propose sont assez nombreuses :

- création, modification, suppression de tables (avec toutes les options) ;
- ajout, modification, suppression des données ;
- possibilité d'entrer des requêtes SQL ;
- importation, exportation de données ;
- *reporting* sur l'état des tables.

Certaines fonctionnalités supplémentaires sont proposées à l'administrateur de la base :

- gestion des droits des utilisateurs ;
- *reload* de la base ;
- création de bases ;
- accès aux données système ainsi qu'aux statistiques ;
- déplacement d'une table d'une base dans une autre.

Cet outil est devenu un atout pour les développeurs web car il permet de gagner énormément de temps. Plutôt que de créer des scripts pour chaque action que vous voulez réaliser sur la table (création, ajout de données, modification de colonnes), phpMyAdmin vous permet de tout faire simplement, et en ligne. Il donne, de plus, accès à des fonctionnalités parfois inconnues ou difficiles à mettre en place.

Faisons un petit tour de ces fonctionnalités…

1 La page d'accueil permet de créer une nouvelle base.



Figure 10.12 : Page d'accueil de phpMyAdmin

2 Par défaut, votre base est vide. En cliquant sur son nom, à gauche, vous avez la possibilité de créer une nouvelle table.

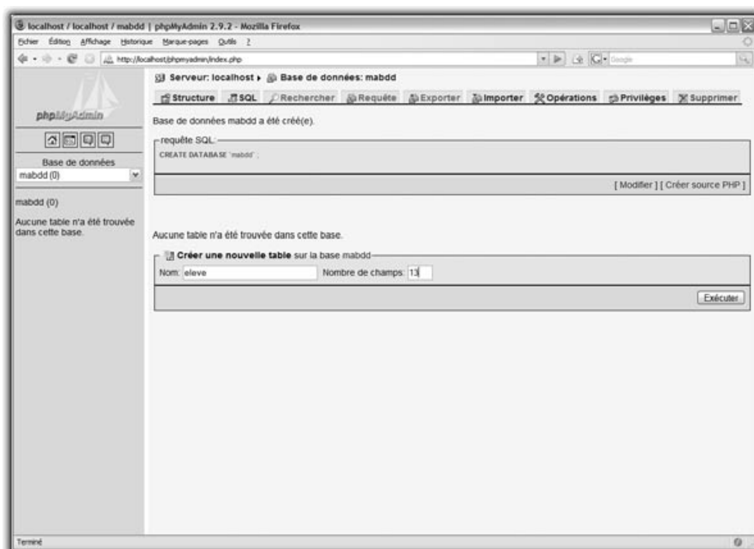


Figure 10.13 : *Création d'une nouvelle table*

3 Vous arrivez ensuite sur une page qui vous permet de préciser les noms et les types de chacune des colonnes. C'est aussi à cet endroit que vous allez spécifier les clés et les index.

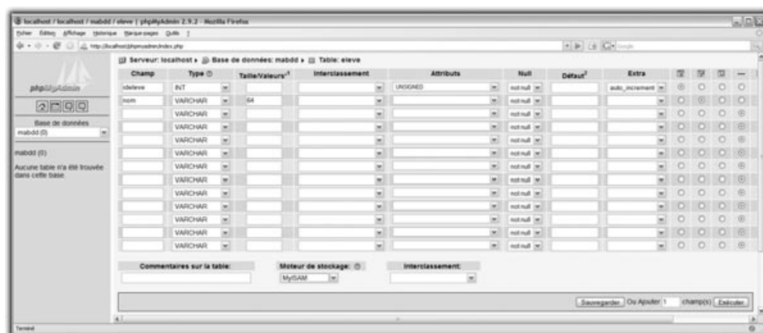


Figure 10.14 : *Paramétrage de la table*

4 Votre table est alors créée. Dans la partie gauche, le nom de la table apparaît en dessous de la base. En cliquant sur le nom de cette table, vous arrivez sur une page qui vous permet de modifier les champs.



Figure 10.15 : *Mise à jour des champs de la table*

5 L'onglet **Insérer** vous permet d'ajouter des données dans la table.



Figure 10.16 : *Insertion dans la table*

6 Une fois la première donnée ajoutée, vous pouvez voir le contenu de la table en cliquant sur la petite icône qui précède le nom de la table dans la zone de gauche.



Figure 10.17 : *Affichage du contenu de la table*

7 L'onglet **SQL** permet de saisir des requêtes SQL

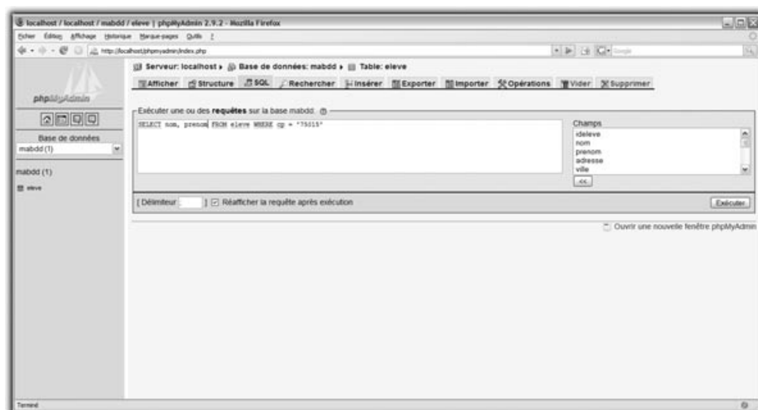


Figure 10.18 : *Saisie d'une requête*

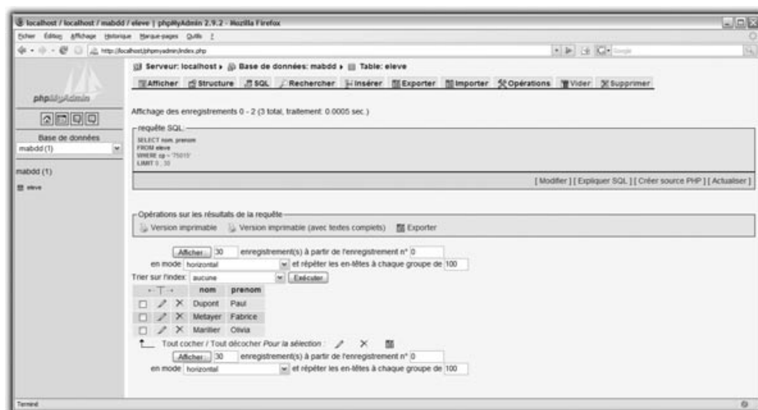


Figure 10.19 : *Résultat de la requête*

8 L'onglet **Exporter** permet d'exporter des données dans différents formats

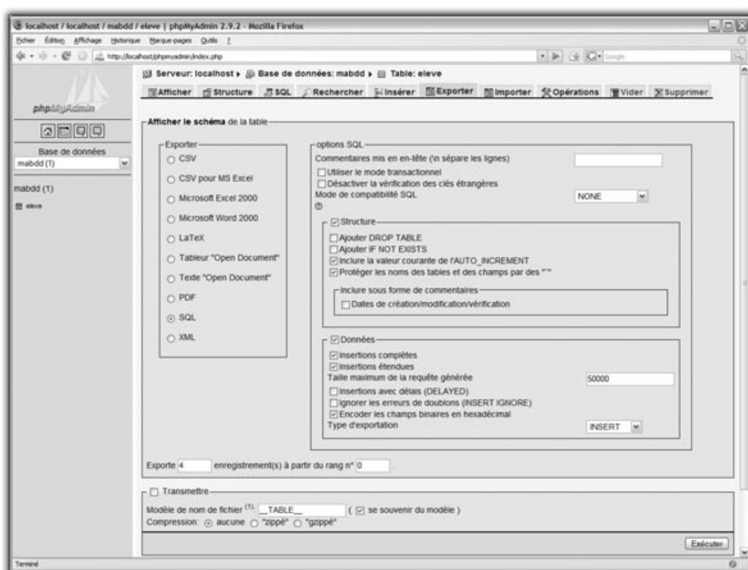


Figure 10.20 : *Exportation des données*

9 Quand des tables existent dans la base, vous obtenez des informations en cliquant sur le nom de cette base. Vous pouvez notamment voir la taille prise par les données contenues dans chaque table.

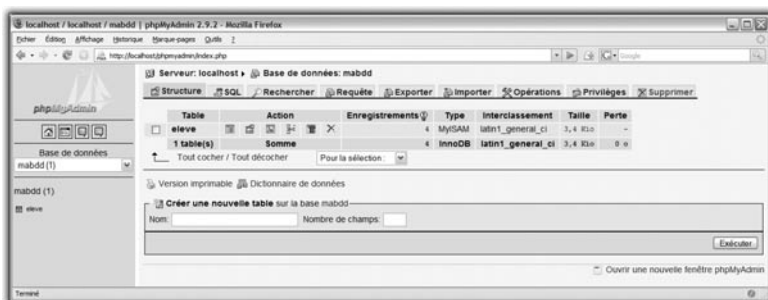


Figure 10.21 : *Votre table contient quatre lignes, représentant 3,4 ko*

Les hébergeurs offrent généralement ce service dès qu'ils proposent la paire PHP/MySQL. Si ce n'est pas le cas, essayez de leur demander. L'installation est réellement aisée, et c'est gratuit.

10.5. Check-list

- MySQL est aujourd'hui considéré comme l'un des meilleurs SGBD du marché.
- Une base de données vise avant tout à stocker et à récupérer des informations le plus vite possible.
- Les SGBD modernes fonctionnent sur un modèle client-serveur.
- SQL est le langage utilisé pour communiquer avec un serveur de bases de données.
- Un SGBD peut contenir plusieurs bases qui contiendront elles-mêmes plusieurs tables.
- Chaque table est définie par un certain nombre de colonnes qui ont chacune un type (numérique, date, chaîne de caractères, etc.). Chaque ligne dans une table est appelée « un enregistrement ».
- Trois étapes sont nécessaires pour se connecter à un SGBD en PHP : la connexion au serveur, la sélection de la base, la transmission de la requête SQL. L'accès à une base nécessite donc quatre éléments : l'adresse du serveur, les identifiants (*login*, *password*) et le nom de la base.
- Les requêtes de type `SELECT` permettent de récupérer de l'information.
- Les requêtes de type `INSERT` permettent d'insérer des informations dans la base.
- La transmission d'un fichier en PHP via un formulaire nécessite que la balise `FORM` dispose des attributs `method="post"` et `enctype="multipart/form-data"`.
- La variable `$_FILES` contient les informations relatives au fichier transmis.
- Le logiciel phpMyAdmin permet de gérer intégralement en ligne un serveur MySQL.

Chapitre 11

La gestion d'une base de données

L'authentification 292
La mise à jour d'une table 296
La suppression : DELETE 308
La factorisation du code 314
Recherche et tri au sein d'une base 331
Check-list 337

Vous allez découvrir dans ce chapitre comment administrer votre base depuis une interface web. Vous utiliserez pour cela intensivement l'ensemble des commandes SQL décrites dans le chapitre précédent.

11.1. L'authentification

Le Web, pour proposer toujours plus de dynamisme et d'interactivité, se complexifie. L'utilisation de scripts et de bases de données est aujourd'hui devenue une quasi-obligation lors de la conception d'un site d'envergure. Une autre tendance semble se dessiner : la présence d'un site caché accompagnant le site principal, et permettant de l'administrer. L'usage est aujourd'hui de parler de *front-office* pour le site visible par un internaute lambda, et de *back-office* pour qualifier le site d'administration, uniquement accessible par le propriétaire du site.

Vous allez donc vous doter, dans ce chapitre, d'un back-office, qui vous permettra de :

- lister les élèves ;
- modifier leur profil ;
- supprimer des élèves ;
- chercher un élève ;
- trier les élèves.

L'accès à ce back-office doit bien évidemment être protégé. Le système d'authentification du protocole HTTP pourra tout 0 fait convenir. Les administrateurs devront renseigner l'identifiant `essai` et le mot de passe `essai` pour se connecter.

La page d'accueil contient la liste des élèves et permet d'accéder à un profil donné :

Listing 11-1 : admin.php

```
<?php

if (!($_SERVER['PHP_AUTH_USER']=="essai" &&
      $_SERVER['PHP_AUTH_PW']=="essai"))
{
    header("status: 401 Unauthorized");
    header("HTTP/1.0 401 Unauthorized");
    header('WWW-authenticate: Basic '.
           ' realm="acces securise au back-office"');
```

```
    print("verification : ERREUR");
    exit(0);
}

echo "<html>";
echo "<head>";
echo "<title>Admin Ecole</title>";
echo "</head>";
echo "<body>";

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");

$sql = "SELECT * FROM eleve";
$resultat = mysql_query ($sql);

echo "<h1>admin - ecole</h1>";

echo "<p align=left> :: accueil</p>";

echo "<table width=90% align=center border=1>";
echo "<tr><td>id</td><td>nom</td><td>prenom</td>
<td>naissance</td><td> </td></tr>";

while ($eleve = mysql_fetch_array ($resultat))
{
    $id = $eleve['ideleve'];
    $nom = $eleve['nom'];
    $prenom = $eleve['prenom'];
    $date = $eleve['naissance'];
    echo "<tr>";
    echo "<td>$id</td>";
    echo "<td>$nom</td>";
    echo "<td>$prenom</td>";
    echo "<td>$date</td>";
    echo "<td>";
    echo "<form action='eleve_edite.php'>";
    echo "<input type='hidden' name='id' value='$id' />";
    echo "<input type='submit' value='voir' />";
    echo "</form>";
    echo "</td>";
    echo "</tr>";
}

echo "</table>";

mysql_close($liendb);

echo "</body>";
echo "</html>";

?>
```

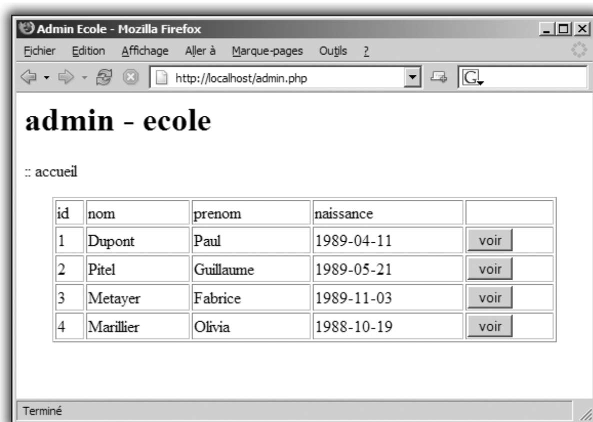


Figure 11.1 : *Page d'accueil*

Écrivez dans la foulée une version allégée du script *eleve_edite.php*, qui est utilisé par *admin.php* afin d'afficher le profil d'un élève :

Listing 11-2 : eleve_edite.php

```
<?php

echo "<html>";
echo "<head>";
echo "<title>Admin Ecole</title>";
echo "</head>";
echo "<body>";

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");

echo "<h1>admin - ecole</h1>";
echo "<p align=left> :: fiche d'eleve
✂ [".$_REQUEST['id']."]</p>";

$sql = "SELECT * FROM eleve WHERE ideleve =
✂ '".$_REQUEST['id']."'";
$resultat = mysql_query ($sql);
$eleve = mysql_fetch_array ($resultat);

echo "-> " . $eleve['nom'] . "<br/>";

mysql_close($liendb);

echo "</body>";
echo "</html>";

?>
```



Figure 11.2 : Visualisation d'un élève

Vous remarquez que, pour visualiser le profil d'un élève, vous avez généré un formulaire par ligne dans le tableau. Si vous cliquez sur un bouton **voir**, vous vous apercevez que l'URL qui apparaît dans la barre d'adresse est de la forme **http://localhost/eleve_edite.php?id=2**.

Ce formulaire intermédiaire peut apparaître quelque peu superflu. Il s'agit d'un exemple typique où l'utilisation de la *Query String* est conseillée. Modifiez la boucle en ce sens :



Figure 11.3 : Utilisation d'un lien pour transmettre de l'information

```
while  ($eleve  =  mysql_fetch_array  ($resultat))
{
    $id  =  $eleve['ideleve'];
    $nom  =  $eleve['nom'];
    $prenom  =  $eleve['prenom'];
    $date  =  $eleve['naissance'];
    echo  "<tr>";
    echo  "<td>$id</td>";
    echo  "<td>$nom</td>";
    echo  "<td>$prenom</td>";
    echo  "<td>$date</td>";
    echo  "<td>";
    echo  "<a href='eleve_edite.php?id=$id'>voir</a>";
    echo  "</td>";
    echo  "</tr>";
}
```

11.2. La mise à jour d’une table

Votre fichier *eleve_edite.php* est pour l’instant très sommaire. Cette partie va vous permettre de le compléter par :

- l’affichage de toutes les données de l’élève dans un formulaire ;
- l’enregistrement des données modifiées.

Commencez par faire en sorte qu’il affiche les données :

Listing 11-3 : Script avec affichage des données

```
<?php

echo "<html>";
echo "<head>";
echo "<title>Admin Ecole</title>";
echo "</head>";
echo "<body>";

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");

echo "<h1>admin - ecole</h1>";
echo "<p align=left> :: fiche d'eleve [".$_REQUEST['id']."]</p>";

$sql = "SELECT * FROM eleve WHERE ideleve =
✂ '".$_REQUEST['id']."'";
$resultat = mysql_query ($sql);
$eleve = mysql_fetch_array ($resultat);

?>
```

```
<form action="eleve_enregistre.php" method="post">
<input type="hidden" name="enregistre" value="oui" />
<input type="hidden" name="id"
       value="<?php echo $_REQUEST['id']; ?>" />

<table>
<tr>
 <td>nom</td>
 <td><input type="text" name="nom"
      value="<?php echo $eleve['nom']; ?>" /></td>
</tr>
<tr>
 <td>prénom</td>
 <td><input type="text" name="prenom"
value="<?php echo $eleve['prenom']; ?>" /></td>
</tr>
<tr>
 <td>adresse</td>
 <td><textarea name="adresse"><?php echo $eleve['adresse']; ?>
 </textarea></td>
</tr>
<tr>
 <td>ville</td>
 <td><input type="text" name="ville"
           value="<?php echo $eleve['ville']; ?>" /></td>
</tr>
<tr>
 <td>code postal</td>
 <td><input type="text" name="codepostal"
           value="<?php echo $eleve['cp']; ?>" /></td>
</tr>
<tr>
 <td>pays</td>
 <td><input type="text" name="pays"
           value="<?php echo $eleve['pays']; ?>" /></td>
</tr>
<tr>
 <td>sexe</td>
 <td>
  M <input type="radio" name="sexe" value="masculin"
  <?php if ($eleve['sexe'] == "masculin") echo "CHECKED"; ?>> -
  F <input type="radio" name="sexe" value="feminin"
  <?php if ($eleve['sexe'] == "feminin") echo "CHECKED"; ?>>
 </td>
</tr>
<tr>
 <td>date naissance</td>
 <td><input type="text" name="naissance"
           value="<?php echo $eleve['naissance']; ?>"
           ✂ /></td>
</tr>
```

```
<tr>
 <td>taille (cm)</td>
 <td><input type="text" name="taille"
            value="<?php echo $eleve['taille']; ?>" /></td>
</tr>
<tr>
 <td>email</td>
 <td><input type="text" name="email"
            value="<?php echo $eleve['email']; ?>" /></td>
</tr>
<tr>
 <td>téléphone</td>
 <td><input type="text" name="telephone"
            value="<?php echo $eleve['telephone']; ?>"
            ✂ /></td>
</tr>
<tr>
 <td>langue vivante</td>
 <td>
   <select name="lv">
     <option value="anglais"> anglais </option>
     <option value="espagnol" <?php if ($eleve['lv'] ==
     ✂ "espagnol")
echo "SELECTED"; ?>> espagnol </option>
     <option value="allemand" <?php if ($eleve['v'] ==
     ✂ "allemand")
echo "SELECTED"; ?>> allemand </option>
   </select>
 </td>
</tr>

</table>

<br/>

<input type="submit" value="enregistrer" />

</form>

</body>
</html>

<?php mysql_close($liendb); ?>
```

Figure 11.4 : *Formulaire prérempli avec les données de l'élève*

Ce script montre comment préremplir un formulaire. Suivant les types de widgets, la méthode n'est pas la même.

- `input text` : la valeur est placée dans l'`input` avec la propriété `value`.

  ```
  value="<?php echo $eleve['nom']; ?>"
  ```

- `textarea` : la valeur est placée entre les balises `<textarea>` et `</textarea>`.
- `input radio` : le terme `CHECKED` doit être placé dans l'`input` valide. Il est donc nécessaire de procéder à un test pour savoir lequel est valide.

  ```
  <?php if ($eleve['sexe'] == "masculin") echo "CHECKED"; ?>
  ```

- `select` : le terme `SELECTED` doit être placé dans l'option valide. Il est donc nécessaire de faire un test pour savoir quelle est la bonne langue vivante.

  ```
  <?php if ($eleve['v'] == "allemand") echo "SELECTED"; ?>
  ```

L’instruction input hidden

Vous remarquez aussi, en dessous de la balise <form>, la présence des deux lignes suivantes :

```
<input  type="hidden"  name="enregistre"  value="oui">
<input  type="hidden"  name="id"
        value="<?php  echo  $id; ?>">
```

Il s’agit d’un `input` particulier qui permet de transmettre une valeur, sans pour autant avoir de répercussion au niveau visuel. Le champ est caché (*hidden*).

Vous avez ajouté le premier champ car vous souhaitez réaliser l’action de mise à jour dans le même fichier : *eleve_edite.php*. Ce champ va permettre de réaliser un test en début de script sur la présence ou non de la variable `$_REQUEST['enregistre']`. Si elle est présente, vous effectuerez l’action de mise à jour, puis vous afficherez le formulaire mis à jour ; sinon, vous vous contenterez d’afficher le formulaire.

Le deuxième champ permet, quant à lui, de propager l’identifiant de l’élève. Si cette ligne n’était pas présente, le script serait incapable de trouver la ligne de la table à mettre à jour, car la valeur `$_REQUEST['id']` ne serait pas transmise.

La commande UPDATE

La nouvelle commande SQL que vous allez utiliser pour mettre à jour la table est `UPDATE`, dont la syntaxe est la suivante :

```
UPDATE nom_table SET nom1 = 'valeur1', nom2 = 'valeur2'...
WHERE condition
```

Dans ce cas, vous avez 12 champs à mettre à jour, et la condition est simple : mettre à jour la ligne ayant pour `ideleve` la valeur `$_REQUEST['id']`. Pour mettre à jour le nom et le prénom, la requête prend la forme suivante :

```
UPDATE  eleve SET nom='".$_REQUEST['nom']."'
WHERE ideleve='".$_REQUEST['id']."'
```



La clause

Si vous n’aviez pas ajouté la clause `WHERE ideleve = '$id'`, c’est l’ensemble de la table qui aurait été mis à jour. Toutes les lignes seraient devenues identiques. De la même manière, vous auriez mis à jour



plusieurs lignes (tous les garçons) si vous aviez choisi une clause du type `WHERE sexe = 'masculin'`.

Écrivez le code permettant la mise à jour :

```
$sql = "UPDATE eleve SET nom = '".$_REQUEST['nom']."',".
       "prenom = '".$_REQUEST['prenom']."',".
       "adresse = '".$_REQUEST['adresse']."',".
       "ville = '".$_REQUEST['ville']."',".
       "cp = '".$_REQUEST['codepostal']."',".
       "pays = '".$_REQUEST['pays']."',".
       "sexe = '".$_REQUEST['sexe']."',".
       "naissance = '".$_REQUEST['naissance']."',".
       "taille = '".$_REQUEST['taille']."',".
       "email = '".$_REQUEST['email']."',".
       "telephone = '".$_REQUEST['telephone']."',".
       "lv = '".$_REQUEST['lv']."'".
       " WHERE ideleve = '".$_REQUEST['id']."'";
```

La mise à jour se fait directement avec les valeurs transmises. Ce n'est pas très judicieux dans la mesure où vous aviez mis en place toute une politique de vérification de champs pour la création.

Rajoutez donc tous ces tests :

```
if ($_REQUEST['enregistre'] == "oui")
{
  if (empty($_REQUEST['nom']) || empty($_REQUEST['prenom']) ||
      empty($_REQUEST['adresse']) ||
      ✂ empty($_REQUEST['ville']) ||
      empty($_REQUEST['codepostal']) ||
      ✂ empty($_REQUEST['pays']) ||
      empty($_REQUEST['naissance']) ||
      ✂ empty($_REQUEST['telephone']) ||
      empty($_REQUEST['lv']))
      die("ERREUR : tous les champs doivent être remplis.");

  if ($_REQUEST['sexe']!="masculin" &&
      $_REQUEST['sexe']!="feminin")
      die("ERREUR : choisissez votre sexe.");

  $reg_email = "/^[\w\.-]+@[\w\.-]+\.[a-z]{2,3}$/i";
  if (preg_match($reg_email,$_REQUEST['email']) == 0)
      die("ERREUR : adresse email non valide.");

  if ($_REQUEST['taille']<=100 || $_REQUEST['taille']>=220)
      die("ERREUR : la taille n'est pas valide.");
```

```
  $sql = "UPDATE eleve SET nom = '".$_REQUEST['nom']."',".
          "prenom = '".$_REQUEST['prenom']."',".
          "adresse = '".$_REQUEST['adresse']."',".
          "ville = '".$_REQUEST['ville']."',".
          "cp = '".$_REQUEST['codepostal']."',".
          "pays = '".$_REQUEST['pays']."',".
          "sexe = '".$_REQUEST['sexe']."',".
          "naissance = '".$_REQUEST['naissance']."',".
          "taille = '".$_REQUEST['taille']."',".
          "email = '".$_REQUEST['email']."',".
          "telephone = '".$_REQUEST['telephone']."',".
          "lv = '".$_REQUEST['lv']."'".
          " WHERE ideleve = '".$_REQUEST['id']."'";

  mysql_query ($sql);
}
```

En rassemblant toutes les parties, votre script *eleve_edite.php* prend la forme suivante :

```
<?php

echo "<html>";
echo "<head>";
echo "<title>Admin Ecole</title>";
echo "</head>";
echo "<body>";

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");

if  ($_REQUEST['enregistre'] == "oui")
{
  if (empty($_REQUEST['nom']) || empty($_REQUEST['prenom']) ||
      empty($_REQUEST['adresse']) ||
      ✂ empty($_REQUEST['ville']) ||
      empty($_REQUEST['codepostal']) ||
      ✂ empty($_REQUEST['pays']) ||
      empty($_REQUEST['naissance']) ||
      ✂ empty($_REQUEST['telephone']) ||
      empty($_REQUEST['lv']))
      die("ERREUR : tous les champs doivent être
      ✂ remplis.");

  if ($_REQUEST['sexe']!="masculin" &&
      $_REQUEST['sexe']!="feminin")
      die("ERREUR : choisissez votre sexe.");

  $reg_email = "/^[\w\.-]+@[\w\.-]+\.[a-z]{2,3}$/i";
  if (preg_match($reg_email,$_REQUEST['email']) == 0)
      die("ERREUR : adresse email non valide.");
```

```
  if ($_REQUEST['taille']<=100 || $_REQUEST['taille']>=220)
      die("ERREUR : la taille n'est pas valide.");

  $sql = "UPDATE eleve SET nom = '".$_REQUEST['nom']."',".
          "prenom = '".$_REQUEST['prenom']."',".
          "adresse = '".$_REQUEST['adresse']."',".
          "ville = '".$_REQUEST['ville']."',".
          "cp = '".$_REQUEST['codepostal']."',".
          "pays = '".$_REQUEST['pays']."',".
          "sexe = '".$_REQUEST['sexe']."',".
          "naissance = '".$_REQUEST['naissance']."',".
          "taille = '".$_REQUEST['taille']."',".
          "email = '".$_REQUEST['email']."',".
          "telephone = '".$_REQUEST['telephone']."',".
          "lv = '".$_REQUEST['lv']."'".
          " WHERE ideleve = '".$_REQUEST['id']."'";

  mysql_query ($sql);
}

echo "<h1>admin - ecole</h1>";
echo "<p align=left> :: fiche d'élève
✂ [".$_REQUEST['id']."]</p>";

$sql = "SELECT * FROM eleve WHERE ideleve =
✂ '".$_REQUEST['id']."'";
$resultat = mysql_query ($sql);
$eleve = mysql_fetch_array ($resultat);

?>
<form action="eleve_edite.php" method="post">
<input type="hidden" name="enregistre" value="oui" />
<input type="hidden" name="id"
       value="<?php echo $_REQUEST['id']; ?>" />

<table>
<tr>
 <td>nom</td>
 <td><input type="text" name="nom"
            value="<?php echo $eleve['nom']; ?>" /></td>
</tr>
<tr>
 <td>prénom</td>
 <td><input type="text" name="prenom"
            value="<?php echo $eleve['prenom']; ?>" /></td>
</tr>
<tr>
 <td>adresse</td>
 <td><textarea name="adresse"><?php echo $eleve['adresse']; ?>
     </textarea></td>
```

```
</tr>
<tr>
 <td>ville</td>
 <td><input type="text" name="ville"
            value="<?php echo $eleve['ville']; ?>" /></td>
</tr>
<tr>
 <td>code postal</td>
 <td><input type="text" name="codepostal"
            value="<?php echo $eleve['cp']; ?>" /></td>
</tr>
<tr>
 <td>pays</td>
 <td><input type="text" name="pays"
            value="<?php echo $eleve['pays']; ?>" /></td>
</tr>
<tr>
 <td>sexe</td>
 <td>
  M <input type="radio" name="sexe" value="masculin"
  <?php if ($eleve['sexe'] == "masculin") echo "CHECKED"; ?>> -
  F <input type="radio" name="sexe" value="feminin"
  <?php if ($eleve['sexe'] == "feminin") echo "CHECKED"; ?>>
 </td>
</tr>
<tr>
 <td>date naissance</td>
 <td><input type="text" name="naissance"
            value="<?php echo $eleve['naissance']; ?>"
            ✂ /></td>
</tr>
<tr>
 <td>taille (cm)</td>
 <td><input type="text" name="taille"
            value="<?php echo $eleve['taille']; ?>" /></td>
</tr>
<tr>
 <td>email</td>
 <td><input type="text" name="email"
            value="<?php echo $eleve['email']; ?>" /></td>
</tr>
<tr>
 <td>téléphone</td>
 <td><input type="text" name="telephone"
            value="<?php echo $eleve['telephone']; ?>" /></td>
</tr>
<tr>
 <td>langue vivante</td>
 <td>
   <select name="lv">
     <option value="anglais"> anglais </option>
```

```
     <option value="espagnol" <?php if ($eleve['lv'] ==
     ✂ "espagnol")
echo "SELECTED"; ?>> espagnol </option>
     <option value="allemand" <?php if ($eleve['v'] ==
     ✂ "allemand")
echo "SELECTED"; ?>> allemand </option>
   </select>
 </td>
</tr>

</table>

<br/>

<input type="submit" value="enregistrer" />

</form>

</body>
</html>

<?php mysql_close($liendb); ?>
```

Le fichier *eleve_edite.php* contient donc à la fois le formulaire et le script. Cette solution peut être intéressante pour diminuer le nombre de fichiers de votre applicatif. Le côté négatif est l’augmentation de la taille du script qui devient ainsi moins lisible.

Appliquez cette technique à la création de profil, en fusionnant les fichiers *ajout_eleve.html* et *eleve_enregistre.php* en un fichier *eleve_ajoute.php* :

```
<?php

if  ($_REQUEST['enregistre']=="oui" && $_REQUEST['id']>=1)
{
  $liendb = mysql_connect("localhost", "root", "");
  mysql_select_db ("test");

  if (empty($_REQUEST['nom']) || empty($_REQUEST['prenom']) ||
      empty($_REQUEST['adresse']) ||
      ✂ empty($_REQUEST['ville']) ||
      empty($_REQUEST['codepostal']) ||
      ✂ empty($_REQUEST['pays']) ||
      empty($_REQUEST['naissance']) ||
      ✂ empty($_REQUEST['telephone']) ||
      empty($_REQUEST['lv']))
      die("ERREUR : tous les champs doivent être
      ✂ remplis.");
```

```
  if ($_REQUEST['sexe']!="masculin" &&
      $_REQUEST['sexe']!="feminin")
      die("ERREUR : choisissez votre sexe.");

  $reg_email = "/^[\w\.-]+@[\w\.-]+\.[a-z]{2,3}$/i";
  if (preg_match($reg_email,$_REQUEST['email']) == 0)
      die("ERREUR : adresse email non valide.");

  if ($_REQUEST['taille']<=100 || $_REQUEST['taille']>=220)
      die("ERREUR : la taille n'est pas valide.");

  $sql = "INSERT INTO eleve (nom, prenom, adresse, ville,
  ✂ cp, pays, sexe, naissance, taille, email, telephone,
  ✂ lv) VALUES (".
          "'".$_REQUEST['nom']."',".
          "'".$_REQUEST['prenom']."',".
          "'".$_REQUEST['adresse']."',".
          "'".$_REQUEST['ville']."',".
          "'".$_REQUEST['codepostal']."',".
          "'".$_REQUEST['pays']."',".
          "'".$_REQUEST['sexe']."',".
          "'".$_REQUEST['naissance']."',".
          "'".$_REQUEST['taille']."',".
          "'".$_REQUEST['email']."',".
          "'".$_REQUEST['telephone']."',".
          "'".$_REQUEST['lv']."')";

  mysql_query ($sql);
  mysql_close($liendb);
  header("location:  http://localhost/admin.php");

}

echo "<html>";
echo "<head>";
echo "<title>Admin Ecole</title>";
echo "</head>";
echo "<body>";

echo "<h1>admin - ecole</h1>";
echo "<p align=left> :: ajouter un élève
✂ [".$_REQUEST['id']."]</p>";

?>

<form action="eleve_ajoute.php" method="post">
<input type="hidden" name="enregistre" value="oui" />

<table>
<tr>
 <td>nom</td>
```

```
 <td><input type="text" name="nom" /></td>
</tr>
<tr>
 <td>prénom</td>
 <td><input type="text" name="prenom" /></td>
</tr>
<tr>
 <td>adresse</td>
 <td><textarea name="adresse"></textarea></td>
</tr>
<tr>
 <td>ville</td>
 <td><input type="text" name="ville" /></td>
</tr>
<tr>
 <td>code postal</td>
 <td><input type="text" name="codepostal" /></td>
</tr>
<tr>
 <td>pays</td>
 <td><input type="text" name="pays" /></td>
</tr>
<tr>
 <td>sexe</td>
 <td>
  M <input type="radio" name="sexe" value="masculin" /> -
  F <input type="radio" name="sexe" value="feminin" />
 </td>
</tr>
<tr>
 <td>date naissance</td>
 <td><input type="text" name="naissance" /></td>
</tr>
<tr>
 <td>taille (cm)</td>
 <td><input type="text" name="taille" /></td>
</tr>
<tr>
 <td>email</td>
 <td><input type="text" name="email" /></td>
</tr>
<tr>
 <td>téléphone</td>
 <td><input type="text" name="telephone" /></td>
</tr>
<tr>
 <td>langue vivante</td>
 <td>
   <select name="lv">
     <option value="anglais"> anglais </option>
     <option value="espagnol"> espagnol </option>
```

```
      <option value="allemand"> allemand </option>
    </select>
  </td>
</tr>

</table>

<br/>

<input type="submit" value="enregistrer" />

</form>

</body>
</html>
```

Après l'ajout à la base, vous basculez sur la liste des élèves. Pour cela, vous transmettez au navigateur la directive HTTP `location` grâce à la fonction PHP `header()`. L'instruction est la suivante :

```
header("location:  http://localhost/admin.php");
```

11.3. La suppression : DELETE

Le fichier *eleve_edite.php* permet déjà de mettre à jour les informations du profil `"élève"`. Vous allez maintenant faire en sorte de pouvoir supprimer un profil d'élève depuis ce fichier.

La commande SQL utilisée pour une suppression d'enregistrement est `DELETE`. La syntaxe est la suivante :

```
DELETE FROM nom_table WHERE clause
```

Si vous souhaitez effacer la ligne dont l'`ideleve` est `4`, écrivez :

```
DELETE  FROM  eleve  WHERE  ideleve  =  '4'
```

Pour pouvoir inclure le code de mise à jour, vous aviez ajouté un `input hidden` qui indiquait au script de réaliser l'action de mise à jour. Deux actions sont désormais possibles : la mise à jour et l'effacement. La technique n'est donc plus valide.

Pour contourner le problème, remplacez le bouton **enregistre** par un menu déroulant et un bouton. Le menu déroulant permettra de choisir l'action à réaliser sur la fiche ; il portera le nom `"action"`. Ainsi, plutôt

que tester la variable `$_REQUEST['enregistre']`, vous allez tester la variable `$_REQUEST['action']` et agir en fonction de sa valeur.

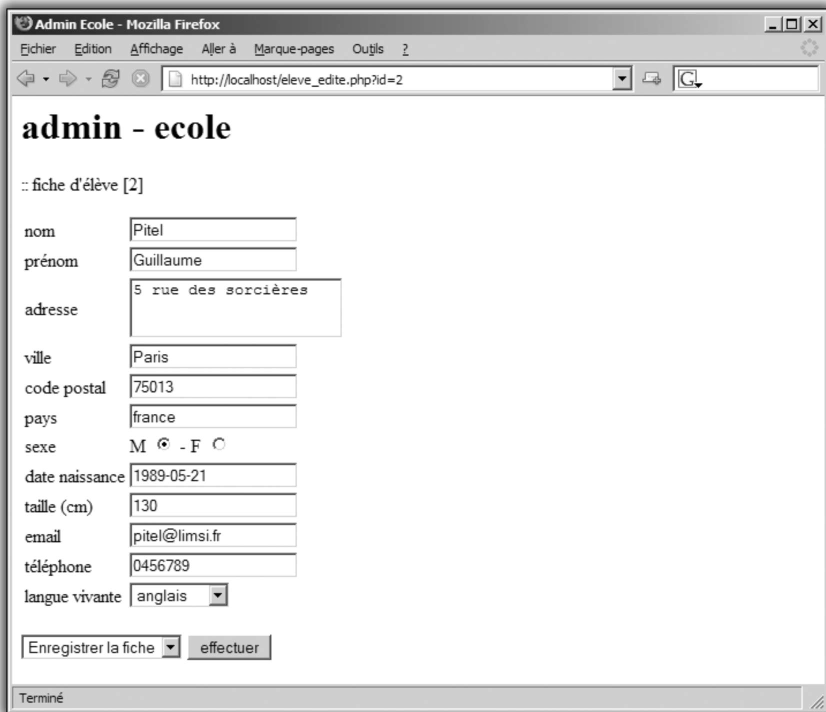


Figure 11.5 : *Mise à jour d'une fiche élève*

Le menu déroulant prend la forme suivante :

```
<select  name="action">
 <option  value="maj">  Enregistrer  la  fiche  </option>
 <option  value="suppr">  Supprimer  la  fiche  </option>
</select>

<input  type="submit"  value="effectuer">
```

Et le test devient :

```
if  ($_REQUEST['action']=="maj")
{
  if (empty($_REQUEST['nom']) || empty($_REQUEST['prenom']) ||
      empty($_REQUEST['adresse']) ||
      ✂ empty($_REQUEST['ville']) ||
      empty($_REQUEST['codepostal']) ||
      ✂ empty($_REQUEST['pays']) ||
```

```
      empty($_REQUEST['naissance']) ||
      ✂ empty($_REQUEST['telephone']) ||
      empty($_REQUEST['lv']))
      die("ERREUR : tous les champs doivent être
      ✂ remplis.");

  if ($_REQUEST['sexe']!="masculin" &&
      $_REQUEST['sexe']!="feminin")
      die("ERREUR : choisissez votre sexe.");

  $reg_email = "/^[\w\.-]+@[\w\.-]+\.[a-z]{2,3}$/i";
  if (preg_match($reg_email,$_REQUEST['email']) == 0)
      die("ERREUR : adresse email non valide.");

  if ($_REQUEST['taille']<=100 || $_REQUEST['taille']>=220)
      die("ERREUR : la taille n'est pas valide.");

  $sql = "UPDATE eleve SET nom = '".$_REQUEST['nom']."',".
          "prenom = '".$_REQUEST['prenom']."',".
          "adresse = '".$_REQUEST['adresse']."',".
          "ville = '".$_REQUEST['ville']."',".
          "cp = '".$_REQUEST['codepostal']."',".
          "pays = '".$_REQUEST['pays']."',".
          "sexe = '".$_REQUEST['sexe']."',".
          "naissance = '".$_REQUEST['naissance']."',".
          "taille = '".$_REQUEST['taille']."',".
          "email = '".$_REQUEST['email']."',".
          "telephone = '".$_REQUEST['telephone']."',".
          "lv = '".$_REQUEST['lv']."'".
          " WHERE ideleve = '".$_REQUEST['id']."'";

  mysql_query ($sql);
}
elseif  ($_REQUEST['action']=="suppr" &&
✂ $_REQUEST['id']>=1)
{
  $sql = "DELETE FROM eleve WHERE
  ✂ ideleve='".$_REQUEST['id']."'";
  mysql_query ($sql);
}
```

Si vous supprimez le profil, vous ne devez surtout pas continuer à exécuter le script, sous peine de générer une erreur. Il est en effet impossible d'afficher un profil d'élève qui n'est plus présent dans la table.

Vous allez donc réutiliser l'instruction suivante :

```
header("location:  http://localhost/admin.php");
```

Avant d'utiliser cette ligne, il est cependant nécessaire de modifier légèrement le script. En effet, vous avez vu dans un chapitre précédent que la fonction `header()` ne pouvait être appelée que si aucun affichage n'avait encore eu lieu. Or, votre script présente les cinq lignes suivantes :

```
echo  "<html>";
echo  "<head>";
echo  "<title>Admin  Ecole</title>";
echo  "</head>";
echo  "<body>";
```

Vous êtes donc contraint de les déplacer plus bas dans le script, au moins en dessous de la fonction `header()`.

Le script final devient donc :

```
<?php

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");

if  ($_REQUEST['action']=="maj")
{
  if (empty($_REQUEST['nom']) || empty($_REQUEST['prenom']) ||
      empty($_REQUEST['adresse']) ||
      ✂ empty($_REQUEST['ville']) ||
      empty($_REQUEST['codepostal']) ||
      ✂ empty($_REQUEST['pays']) ||
      empty($_REQUEST['naissance']) ||
      ✂ empty($_REQUEST['telephone']) ||
      empty($_REQUEST['lv']))
      die("ERREUR : tous les champs doivent être
      ✂ remplis.");

  if ($_REQUEST['sexe']!="masculin" &&
      $_REQUEST['sexe']!="feminin")
      die("ERREUR : choisissez votre sexe.");

  $reg_email = "/^[\w\.-]+@[\w\.-]+\.[a-z]{2,3}$/i";
  if (preg_match($reg_email,$_REQUEST['email']) == 0)
      die("ERREUR : adresse email non valide.");

  if ($_REQUEST['taille']<=100 || $_REQUEST['taille']>=220)
      die("ERREUR : la taille n'est pas valide.");

  $sql = "UPDATE eleve SET nom = '".$_REQUEST['nom']."',".
         "prenom = '".$_REQUEST['prenom']."',".
         "adresse = '".$_REQUEST['adresse']."',".
         "ville = '".$_REQUEST['ville']."',".
```

```
          "cp = '".$_REQUEST['codepostal']."',".
          "pays = '".$_REQUEST['pays']."',".
          "sexe = '".$_REQUEST['sexe']."',".
          "naissance = '".$_REQUEST['naissance']."',".
          "taille = '".$_REQUEST['taille']."',".
          "email = '".$_REQUEST['email']."',".
          "telephone = '".$_REQUEST['telephone']."',".
          "lv = '".$_REQUEST['lv']."'".
          " WHERE ideleve = '".$_REQUEST['id']."'";

  mysql_query ($sql);
}
elseif  ($_REQUEST['action']=="suppr" &&
✂ $_REQUEST['id']>=1)
{
  $sql = "DELETE FROM eleve WHERE
  ✂ ideleve='".$_REQUEST['id']."'";
  mysql_query ($sql);
  header("Location: http://localhost/admin.php");
}

echo "<html>";
echo "<head>";
echo "<title>Admin Ecole</title>";
echo "</head>";
echo "<body>";

echo "<h1>admin - ecole</h1>";
echo "<p align=left> :: fiche d'élève
✂ [".$_REQUEST['id']."]</p>";

$sql = "SELECT * FROM eleve WHERE ideleve =
✂ '".$_REQUEST['id']."'";
$resultat = mysql_query ($sql);
$eleve = mysql_fetch_array ($resultat);

?>

<form action="eleve_edite.php" method="post">
<input type="hidden" name="enregistre" value="oui" />
<input type="hidden" name="id" value="<?php echo
✂ $_REQUEST['id']; ?>" />

<table>
<tr>
 <td>nom</td>
 <td><input type="text" name="nom" value="<?php echo
 ✂ $eleve['nom']; ?>" /></td>
</tr>
<tr>
 <td>prénom</td>
```

```
 <td><input type="text" name="prenom" value="<?php echo
 ✂ $eleve['prenom']; ?>" /></td>
</tr>
<tr>
 <td>adresse</td>
 <td><textarea name="adresse"><?php echo $eleve['adresse'];
 ✂ ?></textarea></td>
</tr>
<tr>
 <td>ville</td>
 <td><input type="text" name="ville" value="<?php echo
 ✂ $eleve['ville']; ?>" /></td>
</tr>
<tr>
 <td>code postal</td>
 <td><input type="text" name="codepostal" value="<?php echo
 ✂ $eleve['cp']; ?>" /></td>
</tr>
<tr>
 <td>pays</td>
 <td><input type="text" name="pays" value="<?php echo
 ✂ $eleve['pays']; ?>" /></td>
</tr>
<tr>
 <td>sexe</td>
 <td>
  M <input type="radio" name="sexe" value="masculin"
  <?php if ($eleve['sexe'] == "masculin") echo "CHECKED"; ?>> -
  F <input type="radio" name="sexe" value="feminin"
  <?php if ($eleve['sexe'] == "feminin") echo "CHECKED"; ?>>
 </td>
</tr>
<tr>
 <td>date naissance</td>
 <td><input type="text" name="naissance" value="<?php echo
 ✂ $eleve['naissance']; ?>" /></td>
</tr>
<tr>
 <td>taille (cm)</td>
 <td><input type="text" name="taille" value="<?php echo
 ✂ $eleve['taille']; ?>" /></td>
</tr>
<tr>
 <td>email</td>
 <td><input type="text" name="email" value="<?php echo
 ✂ $eleve['email']; ?>" /></td>
</tr>
<tr>
 <td>téléphone</td>
 <td><input type="text" name="telephone" value="<?php echo
 ✂ $eleve['telephone']; ?>" /></td>
```

```
</tr>
<tr>
 <td>langue vivante</td>
 <td>
   <select name="lv">
     <option value="anglais"> anglais </option>
     <option value="espagnol" <?php if ($eleve['lv'] ==
     ✂ "espagnol")
echo "SELECTED"; ?>> espagnol </option>
     <option value="allemand" <?php if ($eleve['v'] ==
     ✂ "allemand")
echo "SELECTED"; ?>> allemand </option>
   </select>
 </td>
</tr>

</table>

<br/>

<select  name="action">
 <option  value="maj">  Enregistrer  la  fiche  </option>
 <option  value="suppr">  Supprimer  la  fiche  </option>
</select>

<input  type="submit"  value="effectuer">

</form>

</body>
</html>

<?php mysql_close($liendb); ?>
```



Fonction mysql_close()

Il est préférable de toujours fermer la connexion à la base. Vous quittez le script prématurément si l'action correspond à une suppression. Vous en profitez pour fermer la connexion juste avant de réaliser la redirection (`header()`).

11.4. La factorisation du code

Si vous laissez les scripts *eleve_edite.php* et *eleve_ajoute.php* comme ils sont, vous pouvez avoir des craintes quant à la validité de vos données. Il est en effet nécessaire de s'identifier pour avoir accès à la page d'accueil contenant la liste des élèves, mais aucune vérification n'est

faite sur l'ajout ou la modification (suppression). Toute personne tapant directement `eleve_edite.php?id=6` peut modifier sans problème le contenu de la page.

Il est donc nécessaire de forcer l'identification en haut de chaque script :

```
if (!($_SERVER['PHP_AUTH_USER']=="essai" &&
      $_SERVER['PHP_AUTH_PW']=="essai"))
{
    header("status: 401 Unauthorized");
    header("HTTP/1.0 401 Unauthorized");
    header("WWW-authenticate: Basic realm=\"acces securise\"");
    print("verification : ERREUR");
    exit(0);
}
```

Cette solution est cependant loin d'être optimale. Écrire ce code à plusieurs endroits entraîne une perte de temps et un risque accru de laisser des erreurs. De plus, si vous voulez modifier le code d'accès en `essai2/essai2`, vous serez obligé d'aller modifier tous les fichiers, ce qui compliquerait la mise à jour.

La fonction include

Pour éviter cet écueil, PHP vous propose la fonction `include()`. Elle permet, depuis un script *A*, d'appeler un fichier *B*. Vous pouvez donc créer un fichier *identification.inc.php*, et l'appeler au début de chaque script devant être protégé :

```
<?php

if (!($_SERVER['PHP_AUTH_USER']=="essai" &&
      $_SERVER['PHP_AUTH_PW']=="essai"))
{
    header("status: 401 Unauthorized");
    header("HTTP/1.0 401 Unauthorized");
    header("WWW-authenticate: Basic realm=\"acces
    ✂ securise\"");
    print("verification : ERREUR");
    exit(0);
}

?>
```

Cette technique permet de modifier la gestion de l'identification en un instant, qu'il y ait 5 ou 5000 scripts dans votre applicatif.

Dans la même veine, vous pouvez aussi remarquer l'utilisation dans plusieurs scripts d'une connexion à une base de données. En ayant à l'esprit la fonction `include()` et en étant suffisamment paresseux, il vous viendra tout de suite à l'idée de créer le fichier *variables.inc.php* :

Listing 11-4 : variables.inc.php

```
<?php

$bddserver  =  "localhost";
$bddlogin  =  "root";
$bddpassword  =  "";
$bdd  =  "test";
$table_eleve  =  "eleve";
$url  =  "http://localhost";

?>
```

Ce fichier est aussi inclus en haut de tous les scripts et vous permet de déplacer votre applicatif d'un hébergeur vers un autre sans le moindre souci. Tous les paramètres susceptibles de changer sont isolés dans un fichier unique. Stocker de telles variables dans un fichier extérieur est donc une bonne habitude à prendre.

Ne nous arrêtons pas là dans cette quête « factorisatrice » et regardons du côté du visuel… Chaque page dispose du même en-tête et du même pied de page. Comme il y a fort à parier qu'à un moment ou à un autre vous vouliez améliorer visuellement l'ensemble et placer sur chaque page une référence à un copyright, il pourrait donc être tout à fait judicieux de créer les fichiers *haut.inc.php* et *bas.inc.php* :

Listing 11-5 : haut.inc.php

```
<html>
<head>
 <title>Admin Ecole</title>
</head>
<body>

<h1>Admin - Ecole</h1>
```

Listing 11-6 : bas.inc.php

```
</body>
</html>
```

Chaque script du back-office sera donc construit de la façon suivante :

- appel à *var.inc.php* ;

- appel à *identification.inc.php* ;
- appel à *haut.inc.php* ;
- le contenu proprement dit du script ;
- *bas.inc.php*.

Modifiez en ce sens les fichiers *admin.php, eleve_ajoute.php*, et *eleve_edite.php* :

Listing 11-7 : admin.php

```
<?php

include("variables.inc.php");
include("identification.inc.php");
include("haut.inc.php");

$liendb = mysql_connect ($bddserver, $bddlogin, $bddpassword);
mysql_select_db ($bdd);

$sql = "SELECT * FROM eleve";
$resultat = mysql_query ($sql);

echo "<p align=left> :: accueil</p>";

echo "<table width=90% align=center border=1>";
echo "<tr><td>id</td><td>nom</td><td>prenom</td>
<td>naissance</td><td> </td></tr>";

while ($eleve = mysql_fetch_array ($resultat))
{
  $id = $eleve['ideleve'];
  $nom = $eleve['nom'];
  $prenom = $eleve['prenom'];
  $date = $eleve['naissance'];
  echo "<tr>";
  echo "<td>$id</td>";
  echo "<td>$nom</td>";
  echo "<td>$prenom</td>";
  echo "<td>$date</td>";
  echo "<td>";
  echo "<a href='eleve_edite.php?id=$id'>voir</a>";
  echo "</td>";
  echo "</tr>";
}

echo "</table>";

mysql_close($liendb);
```

```
include("bas.inc.php");

?>
```

Listing 11-8 : eleve_ajoute.inc.php

```
<?php

include("variables.inc.php");
include("identification.inc.php");

if  ($_REQUEST['enregistre'] == "oui")
{
  $liendb = mysql_connect ($bddserver, $bddlogin,
  ✄ $bddpassword);
  mysql_select_db ($bdd);

  if (empty($_REQUEST['nom']) || empty($_REQUEST['prenom']) ||
      empty($_REQUEST['adresse']) ||
      ✄ empty($_REQUEST['ville']) ||
      empty($_REQUEST['codepostal']) ||
      ✄ empty($_REQUEST['pays']) ||
      empty($_REQUEST['naissance']) ||
      ✄ empty($_REQUEST['telephone']) ||
      empty($_REQUEST['lv']))
      die("ERREUR : tous les champs doivent être remplis.");

  if ($_REQUEST['sexe']!="masculin" &&
      $_REQUEST['sexe']!="feminin")
      die("ERREUR : choisissez votre sexe.");

  $reg_email = "/^[\w\.-]+@[\w\.-]+\.[a-z]{2,3}$/i";
  if (preg_match($reg_email,$_REQUEST['email']) == 0)
      die("ERREUR : adresse email non valide.");

  if ($_REQUEST['taille']<=100 || $_REQUEST['taille']>=220)
      die("ERREUR : la taille n'est pas valide.");

  $sql = "INSERT INTO eleve (nom, prenom, adresse, ville,
  ✄ cp, pays, sexe, naissance, taille, email, telephone,
  ✄ lv) VALUES (".
          "'".$_REQUEST['nom']."',".
          "'".$_REQUEST['prenom']."',".
          "'".$_REQUEST['adresse']."',".
          "'".$_REQUEST['ville']."',".
          "'".$_REQUEST['codepostal']."',".
          "'".$_REQUEST['pays']."',".
          "'".$_REQUEST['sexe']."',".
          "'".$_REQUEST['naissance']."',".
          "'".$_REQUEST['taille']."',".
```

```
          "'".$_REQUEST['email']."',".
          "'".$_REQUEST['telephone']."',".
          "'".$_REQUEST['lv']."')";

    mysql_query ($sql);
    mysql_close($liendb);
    header("location:  http://localhost/admin.php");

}

include("haut.inc.php");

echo "<p align=left> :: ajouter un élève
✂ [".$_REQUEST['id']."]</p>";

?>

<form action="eleve_ajoute.php" method="post">
<input type="hidden" name="enregistre" value="oui" />

<table>
<tr>
 <td>nom</td>
 <td><input type="text" name="nom" /></td>
</tr>
<tr>
 <td>prénom</td>
 <td><input type="text" name="prenom" /></td>
</tr>
<tr>
 <td>adresse</td>
 <td><textarea name="adresse"></textarea></td>
</tr>
<tr>
 <td>ville</td>
 <td><input type="text" name="ville" /></td>
</tr>
<tr>
 <td>code postal</td>
 <td><input type="text" name="codepostal" /></td>
</tr>
<tr>
 <td>pays</td>
 <td><input type="text" name="pays" /></td>
</tr>
<tr>
 <td>sexe</td>
 <td>
  M <input type="radio" name="sexe" value="masculin" /> -
  F <input type="radio" name="sexe" value="feminin" />
 </td>
```

```
</tr>
<tr>
 <td>date naissance</td>
 <td><input type="text" name="naissance" /></td>
</tr>
<tr>
 <td>taille (cm)</td>
 <td><input type="text" name="taille" /></td>
</tr>
<tr>
 <td>email</td>
 <td><input type="text" name="email" /></td>
</tr>
<tr>
 <td>téléphone</td>
 <td><input type="text" name="telephone" /></td>
</tr>
<tr>
 <td>langue vivante</td>
 <td>
   <select name="lv">
     <option value="anglais"> anglais </option>
     <option value="espagnol"> espagnol </option>
     <option value="allemand"> allemand </option>
   </select>
 </td>
</tr>

</table>

<br/>

<input type="submit" value="enregistrer" />

</form>

<?php include ("bas.inc.php"); ?>
```

Listing 11-9 : eleve_edite.php

```
<?php

include ("variables.inc.php");
include ("identification.inc.php");

$liendb = mysql_connect($bddserver, $bddlogin, $bddpassword);
mysql_select_db  ($bdd);

if  ($_REQUEST['action']=="maj")
{
  if (empty($_REQUEST['nom']) || empty($_REQUEST['prenom']) ||
```

```
        empty($_REQUEST['adresse']) ||
        ✂ empty($_REQUEST['ville']) ||
        empty($_REQUEST['codepostal']) ||
        ✂ empty($_REQUEST['pays']) ||
        empty($_REQUEST['naissance']) ||
        ✂ empty($_REQUEST['telephone']) ||
        empty($_REQUEST['lv']))
        die("ERREUR : tous les champs doivent être
        ✂ remplis.");

    if ($_REQUEST['sexe']!="masculin" &&
        $_REQUEST['sexe']!="feminin")
        die("ERREUR : choisissez votre sexe.");

    $reg_email = "/^[\w\.-]+@[\w\.-]+\.[a-z]{2,3}$/i";
    if (preg_match($reg_email,$_REQUEST['email']) == 0)
        die("ERREUR : adresse email non valide.");

    if ($_REQUEST['taille']<=100 || $_REQUEST['taille']>=220)
        die("ERREUR : la taille n'est pas valide.");

    $sql = "UPDATE eleve SET nom = '".$_REQUEST['nom']."',".
           "prenom = '".$_REQUEST['prenom']."',".
           "adresse = '".$_REQUEST['adresse']."',".
           "ville = '".$_REQUEST['ville']."',".
           "cp = '".$_REQUEST['codepostal']."',".
           "pays = '".$_REQUEST['pays']."',".
           "sexe = '".$_REQUEST['sexe']."',".
           "naissance = '".$_REQUEST['naissance']."',".
           "taille = '".$_REQUEST['taille']."',".
           "email = '".$_REQUEST['email']."',".
           "telephone = '".$_REQUEST['telephone']."',".
           "lv = '".$_REQUEST['lv']."'".
           " WHERE ideleve = '".$_REQUEST['id']."'";

    mysql_query ($sql);
  }
  elseif  ($_REQUEST['action']=="suppr" &&
  ✂ $_REQUEST['id']>=1)
  {
    $sql = "DELETE FROM eleve WHERE
    ✂ ideleve='".$_REQUEST['id']."'";
    mysql_query ($sql);
    header("Location: http://localhost/admin.php");
  }

  include("haut.inc.php");

  echo "<p align=left> :: fiche d'élève
  ✂ [".$_REQUEST['id']."]</p>";
```

```
$sql = "SELECT * FROM eleve WHERE ideleve =
✂ '".$_REQUEST['id']."'";
$resultat = mysql_query ($sql);
$eleve = mysql_fetch_array ($resultat);

?>

<form action="eleve_edite.php" method="post">
<input type="hidden" name="enregistre" value="oui" />
<input type="hidden" name="id" value="<?php echo
✂ $_REQUEST['id']; ?>" />

<table>
<tr>
 <td>nom</td>
 <td><input type="text" name="nom" value="<?php echo
 ✂ $eleve['nom']; ?>" /></td>
</tr>
<tr>
 <td>prénom</td>
 <td><input type="text" name="prenom" value="<?php echo
 ✂ $eleve['prenom']; ?>" /></td>
</tr>
<tr>
 <td>adresse</td>
 <td><textarea name="adresse"><?php echo $eleve['adresse'];
 ✂ ?></textarea></td>
</tr>
<tr>
 <td>ville</td>
 <td><input type="text" name="ville" value="<?php echo
 ✂ $eleve['ville']; ?>" /></td>
</tr>
<tr>
 <td>code postal</td>
 <td><input type="text" name="codepostal" value="<?php echo
 ✂ $eleve['cp']; ?>" /></td>
</tr>
<tr>
 <td>pays</td>
 <td><input type="text" name="pays" value="<?php echo
 ✂ $eleve['pays']; ?>" /></td>
</tr>
<tr>
 <td>sexe</td>
 <td>
  M <input type="radio" name="sexe" value="masculin"
  <?php if ($eleve['sexe'] == "masculin") echo "CHECKED"; ?>> -
  F <input type="radio" name="sexe" value="feminin"
  <?php if ($eleve['sexe'] == "feminin") echo "CHECKED"; ?>>
 </td>
```

```
</tr>
<tr>
 <td>date naissance</td>
 <td><input type="text" name="naissance" value="<?php echo
 ✂ $eleve['naissance']; ?>" /></td>
</tr>
<tr>
 <td>taille (cm)</td>
 <td><input type="text" name="taille" value="<?php echo
 ✂ $eleve['taille']; ?>" /></td>
</tr>
<tr>
 <td>email</td>
 <td><input type="text" name="email" value="<?php echo
 ✂ $eleve['email']; ?>" /></td>
</tr>
<tr>
 <td>téléphone</td>
 <td><input type="text" name="telephone" value="<?php echo
 ✂ $eleve['telephone']; ?>" /></td>
</tr>
<tr>
 <td>langue vivante</td>
 <td>
   <select name="lv">
     <option value="anglais"> anglais </option>
     <option value="espagnol" <?php if ($eleve['lv'] ==
     ✂ "espagnol")
echo "SELECTED"; ?>> espagnol </option>
     <option value="allemand" <?php if ($eleve['v'] ==
     ✂ "allemand")
echo "SELECTED"; ?>> allemand </option>
   </select>
 </td>
</tr>

</table>

<br/>

<select  name="action">
 <option  value="maj">  Enregistrer  la  fiche  </option>
 <option  value="suppr">  Supprimer  la  fiche  </option>
</select>

<input  type="submit"  value="effectuer">

</form>

</body>
</html>
```

```
<?php

mysql_close ($liendb);
include ("bas.inc.php");

?>
```

Modifiez, tout de suite, les fichiers *haut.inc.php* et *bas.inc.php* pour vous rendre compte de l'intérêt de cette action :

```
<html>
<head>
 <title>Admin Ecole</title>
</head>
<body>

<h1>Admin - Ecole</h1>

<center>

<hr/>
<br/><br/>
<hr/>

<a href="admin.php">accueil</a> -
<a href="eleve_ajoute.php">ajouter un  élève</a>

<hr>

© libre

</center>

</body>
</html>
```

Instantanément, tous vos scripts disposent de la nouvelle mise en page. (voir Figure 9.6)

En conclusion, il est très important de toujours avoir à l'esprit la factorisation de votre code :

- Cela le rend plus lisible, plus facile à maintenir, à mettre à jour, à faire évoluer.
- Cela vous fait gagner du temps.
- Cela réduit la taille du code et ainsi la présence de bugs.

Cette technique est aussi de plus en plus utilisée en front-office, et porte le nom de *template*. Ces templates permettent de modifier très rapidement l'intégralité du site.

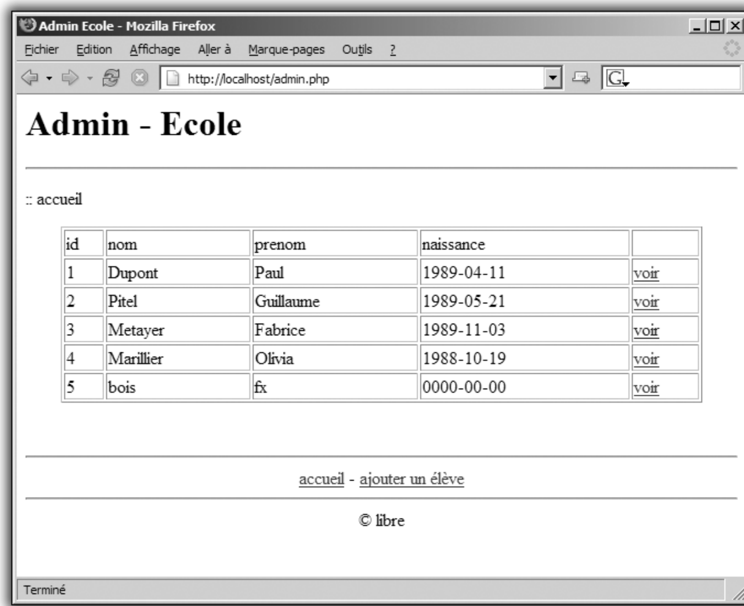


Figure 11.6 : *Nouvelle mise en page*

L'amélioration visuelle : les CSS

Avant d'enrichir votre applicatif d'autres fonctionnalités, nous allons nous arrêter quelques instants sur son aspect visuel. Bien que cela semble souvent secondaire, bien présenter une page, en jouant sur les couleurs, les polices, etc., peut faciliter considérablement la lisibilité et la compréhension de votre page.

Le HTML donne la possibilité de jouer sur les polices, les couleurs, les fonds… La première solution qui vient donc à l'esprit est de modifier directement le code afin d'inclure des indications visuelles. Pour avoir un tableau avec un fond bleu, vous pouvez écrire :

```
<table  bgcolor="blue">
```

Cependant, c'est encore une fois une mauvaise approche car elle n'a pas l'avantage de factoriser le travail. En effet, si vous souhaitez modifier la couleur du titre des rubriques, il faut revenir sur tous les scripts et les modifier un à un.

Pour faciliter la gestion de l'aspect visuel de votre applicatif, vous allez tirer partie des CSS (*Cascading Style Sheets*). Ce sont des commandes qui permettent, là encore, de modifier l'aspect visuel des pages, et qui ont un double avantage :

- Vous pouvez apporter des modifications plus fines (taille, espacement de lettres, etc.) et plus nombreuses (boutons, menus, etc.).
- Vous pouvez gérer, en un seul endroit, les attributs visuels de toutes les pages.

Les instructions CSS peuvent être placées :

- directement dans les balises ;

```
<p style="background: #CCCCCC; color: black;">
```

- dans l'en-tête de la page.

```
<html>
<head>
<style type="text/css">
<!--
P {background: #CCCCCC; color: black;}
-->
</style>
</head>
</html>
```

Grâce à votre factorisation PHP et à l'existence de votre fichier *haut.inc.php*, vous allez être en mesure de modifier l'intégralité des scripts en n'intervenant que sur un fichier :

```
<html>
<head>
<title>Admin Ecole</title>
<style type="text/css">
<!--
BODY {background: #D0D8D5;}
-->
</style>
</head>
<body>
<h1>Admin - Ecole</h1>
<center>
<hr>
```

Avec cette modification, toutes vos pages ont désormais un fond vert très clair.



Figure 11.7 : *Modification du fond de la page*

Sur le même modèle, il est possible de modifier l'intégralité des éléments HTML de vos pages :

```
<html>
<head>
<title>Admin Ecole</title>
<style type="text/css">
<!--

BODY
{background:#D0D8D5; color:#01291A; font-family:Verdana;
✂ font-size:10px;}

H1
{background:#0B3E2B; color:white; font-family:Arial;
✂ font-size:20px; font-weight:bold;}

P
{background:#B6CCC4; color:#0B3E2B; font-family:Arial;
✂ font-size:14px; font-weight:bold;}

TD
```

```
{background:white; color:#0B3E2B; font-family:Arial;
✂ font-size:14px; font-weight:normal;}

INPUT
{background:#EBF0EE; color:#01291A; font-family:Verdana;
✂ font-size:10px;}

TEXTAREA
{background:#EBF0EE; color:#01291A; font-family:Verdana;
✂ font-size:10px;}

SELECT
{background:#EBF0EE; color:#01291A; font-family:Verdana;
✂ font-size:10px;}

HR {color:#0B3E2B;}

A {color:#AF6114; font-family:verdana; font-size:10px;}

-->
</style>

</head>
<body>
<h1>Admin - Ecole</h1>
<center>
<hr/>
```

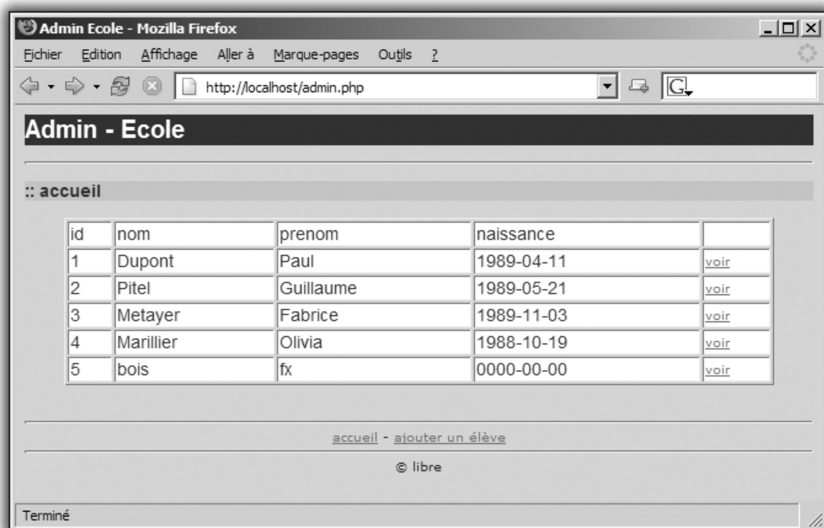


Figure 11.8 : *Encore une nouvelle présentation*

L'aspect devient maintenant plus agréable, mais reste encore légèrement monotone. Toutes les cases de vos tableaux ont un fond blanc. Il serait donc préférable, dans le fichier *admin.php*, de démarquer la première ligne, qui contient les intitulés des colonnes. Pour cela, il est nécessaire d'intervenir dans le fichier et d'ajouter une propriété à tous les `<td>` en question. La propriété est `class='nom_de_la_class'`. Si vous choisissez de l'appeler `intitule`, vous modifiez *admin.php* de cette manière :

```
echo "<table width=90% align=center border=1>";
echo "<tr>";
echo "<td class='intitule'>id</td>";
echo "<td class='intitule'>nom</td>";
echo "<td class='intitule'>prenom</td>";
echo "<td class='intitule'>naissance</td>";
echo "<td class='intitule'> </td>";
echo "</tr>";
```

La présence de cette classe va vous permettre d'affiner votre CSS et de faire en sorte que les cellules de tableaux ayant la classe `intitule` aient un aspect visuel différent. Précisez-le avec la notation `TD.intitule` :

```
<html>
<head>
<title>Admin Ecole</title>
<style type="text/css">
<!--

BODY
{background:#D0D8D5; color:#01291A; font-family:Verdana;
✂ font-size:10px;}

H1
{background:#0B3E2B; color:white; font-family:Arial;
✂ font-size:20px; font-weight:bold;}

P
{background:#B6CCC4; color:#0B3E2B; font-family:Arial;
✂ font-size:14px; font-weight:bold;}

TD
{background:white; color:#0B3E2B; font-family:Arial;
✂ font-size:14px; font-weight:normal;}

TD.intitule
{background:#777; color:white; font-family:Arial;
✂ font-size:14px; font-weight:bold;}

INPUT
```

```
{background:#EBF0EE; color:#01291A; font-family:Verdana;
✂ font-size:10px;}

TEXTAREA
{background:#EBF0EE; color:#01291A; font-family:Verdana;
✂ font-size:10px;}

SELECT
{background:#EBF0EE; color:#01291A; font-family:Verdana;
✂ font-size:10px;}

HR {color:#0B3E2B;}

A {color:#AF6114; font-family:verdana; font-size:10px;}

-->
</style>

</head>
<body>
<h1>Admin - Ecole</h1>
<center>
<hr/>
```

Vous auriez aussi pu écrire :

```
.intitule
{background:#0B3E2B; color:white; font-family:Arial;
✂ font-size:14px; font-weight:bold;}
```

Dans ce cas, toutes les balises ayant la classe `intitule` auraient partagé le même style, par exemple `<p class='intitule'>`.

Une des dernières propriétés intéressantes des CSS est de permettre de modifier le style d’un élément de la page en fonction d’un événement. Ainsi, il est possible de changer l’aspect visuel des liens lorsque vous passez le pointeur de la souris au-dessus :

```
A:hover
{background:#AF6114; color:white; font-family:Verdana;
✂ font-size:10px;}
```

Avant de finir cette partie, il convient de revenir sur quelques inconvénients des CSS :

- Certains navigateurs anciens ne gèrent pas du tout les CSS.
- Selon les navigateurs, certains styles sont absents ou mal traités.

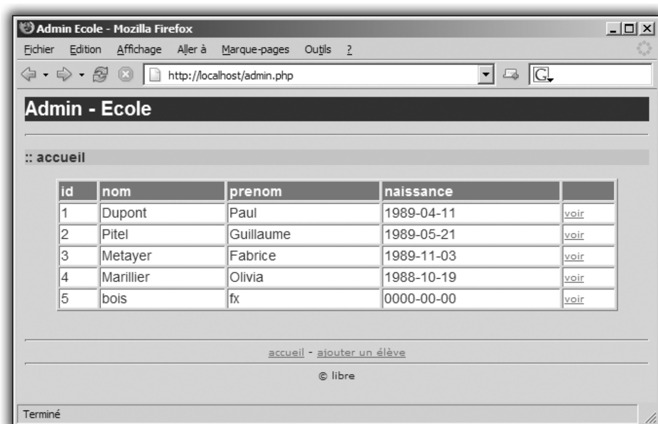


Figure 11.9 : *Nouvelle présentation avec changement de l'aspect visuel des liens*

11.5. Recherche et tri au sein d'une base

La fonction de recherche et de tri d'éléments de la base est souvent utile au sein d'un back-office. Vous allez donc modifier la page d'accueil en lui ajoutant, tout d'abord, un petit formulaire permettant de taper le nom ou le prénom d'un élève, puis un menu qui offrira la possibilité de lister les élèves suivant un critère donné.

Définir la fonction de recherche

La première idée qui peut venir à l'esprit est de créer un nouveau script nommé par exemple *eleve_recherche.php*, qui listerait les élèves correspondant à un mot-clé. Cependant, en y regardant de plus près, vous pouvez vous apercevoir qu'il s'agit en fait du fichier *admin.php*, auquel il suffit d'ajouter une clause à la fin de la requête `SELECT`.

L'idée est donc de générer une clause dès qu'un mot-clé est passé en paramètre. Rien de bien sorcier en perspective ! Le moteur de recherche doit par contre être suffisamment permissif et vous renvoyer `Paul Dupont` même si vous ne tapez que `Dup`. L'utilisation de l'opérateur d'égalité (=) dans la clause n'est donc pas valide. Ce qu'il faut employer, dans ce cas, c'est la fonction MySQL `INSTR()`. Cette

fonction prend deux arguments : la donnée dans laquelle on cherche un motif et le motif en question. Le premier argument est généralement le nom d'une colonne.

La requête est donc construite ainsi :

```
SELECT * FROM $table_eleve WHERE INSTR(nom,$motclef) OR
✂ INSTR(prenom,$motclef)
```



Fonction MySQL

Il existe une multitude de fonctions qui peuvent être incluses directement dans des requêtes SQL. Vous en trouverez un certain nombre dans les annexes.

Listing 11-10 : Moteur de recherche intégré

```
<?php

include("variables.inc.php");
include("identification.inc.php");
include("haut.inc.php");

$liendb = mysql_connect ($bddserver, $bddlogin,
✂ $bddpassword);
mysql_select_db ($bdd);

?>

<p align='left'>:: accueil</p>

<form action="admin.php" method="post">
  <input type="text" name="motclef"
         value="<?php echo $_REQUEST['motclef'];?>" />
  <input type="submit" value="rechercher" />
</form>


<table width='90%' border='1'>

<tr>
 <td class='intitule'>id</td>
 <td class='intitule'>nom</td>
 <td class='intitule'>prenom</td>
 <td class='intitule'>naissance</td>
 <td class='intitule'> </td>
</tr>

<?php
```

```
$clause = '';
if  (isset($_REQUEST['motclef'])) {
  $clause .= " WHERE
  ✂ INSTR(nom,'".$_REQUEST['motclef']."')";
  $clause .= " OR
  ✂ INSTR(prenom,'".$_REQUEST['motclef']."')";
}

$sql = "SELECT * FROM eleve ".$clause;
$resultat = mysql_query ($sql);

while ($eleve = mysql_fetch_array ($resultat))
{
  $id = $eleve['ideleve'];
  $nom = $eleve['nom'];
  $prenom = $eleve['prenom'];
  $date = $eleve['naissance'];
  echo "<tr>";
  echo "<td>$id</td>";
  echo "<td>$nom</td>";
  echo "<td>$prenom</td>";
  echo "<td>$date</td>";
  echo "<td>";
  echo "<a href='eleve_edite.php?id=$id'>voir</a>";
  echo "</td>";
  echo "</tr>";
}

echo "</table>";

mysql_close($liendb);

include("bas.inc.php");

?>
```

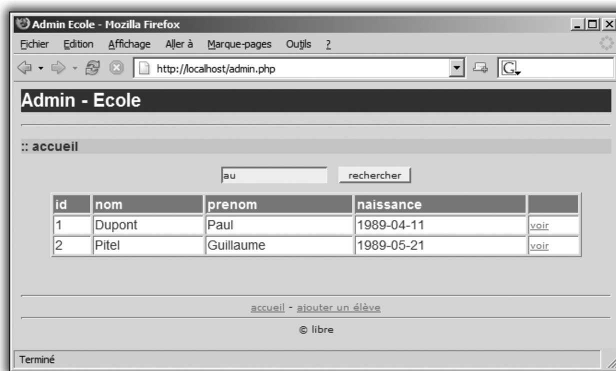


Figure 11.10 : *Résultat de la recherche sur le mot 'au'*

Dans le cas où un mot-clé a été tapé, vous entrez dans la condition `if`, et la variable `$clause` est initialisée, par exemple :

```
" WHERE INSTR(nom,'au') OR INSTR(prenom,'au')")
```

La variable `$sql` est ensuite initialisée. Si la variable `$clause` n’a pas été créée, la requête reste `"SELECT * FROM $table_eleve"`, sinon la clause est ajoutée :

```
"SELECT * FROM $table_eleve WHERE INSTR(nom,'au') OR
✂ INSTR(prenom,'au')"
```

Si vous tapez à présent le mot-clé `dup`, vous n’obtenez aucun résultat car `INSTR` est sensible à la casse.

Pour lister tous les élèves, sans vous préoccuper de la différence minuscules/majuscules, préférez l’opérateur de comparaison `LIKE` :

Listing 11-11 : Test non sensible à la casse

```
if  (isset($motclef))
{
  $clause = "WHERE nom LIKE '".$_REQUEST['motclef']."' OR
  ✂ prenom LIKE '".$_REQUEST['motclef']."'";
}
```

Il est important de ne jamais hésiter à composer vos requêtes à partir d’éléments disparates. Cela permet d’aboutir à des fonctionnalités très intéressantes sans trop d’efforts.

Définir la fonction de tri

L’autre fonctionnalité très utile en SGBD est la possibilité d’obtenir les données triées. Pour illustrer cette fonctionnalité, vous allez ajouter au script *admin.php* la possibilité de trier les élèves par noms, prénoms ou dates de naissance.

Pour préciser un ordre, il suffit d’ajouter la commande `ORDER BY` à la fin de la requête. Ainsi, si vous souhaitez classer les élèves par noms, écrivez :

```
SELECT * FROM eleve ORDER BY nom
```

Il est possible de réaliser un classement multiple en définissant un `ORDER BY` sur plusieurs colonnes. Si vous souhaitez classer les élèves par noms et, pour un même nom, par dates de naissance, écrivez :

```
ORDER BY nom, naissance
```



ORDER BY et WHERE

Les commandes ORDER BY et WHERE ne sont pas exclusives. Les deux peuvent cohabiter. Il est cependant nécessaire de mettre le WHERE avant l'ORDER BY.

Tout comme pour la recherche, vous voyez qu'il suffit d'enrichir la requête si une variable $ordre est transmise :

```
<?php

include("variables.inc.php");
include("identification.inc.php");
include("haut.inc.php");

$liendb = mysql_connect($bddserver, $bddlogin,
✂ $bddpassword);
mysql_select_db ($bdd);

?>

<p align="left"> :: accueil</p>

<form action="admin.php" method="post">
<input type="text" name="motclef" value="<?php echo
✂ $motclef; ?>"> <input type="submit"
✂ value="rechercher">
</form>

<table width="90%" align="center" border="1">
  <tr>
    <td class="intitule">id</td>
    <td class="intitule">nom</td>
    <td class="intitule">prenom</td>
    <td class="intitule">naissance</td>
    <td class="intitule"> </td>
  </tr>


<?php

if (isset($motclef))
{
    $clause = " WHERE nom LIKE '%$motclef%' OR
    ✂ prenom LIKE '%$motclef%'";
}

if (isset($ordre))
{
    $orderby = " ORDER BY $ordre";
}
```

```
$sql = "SELECT * FROM $table_eleve" . $clause .
✂ $orderby;
$resultat = mysql_query ($sql);

while ($eleve = mysql_fetch_array ($resultat))
{
    $id = $eleve['ideleve'];
    $nom = $eleve['nom'];
    $prenom = $eleve['prenom'];
    $date = $eleve['naissance'];
    echo "<tr>";
    echo "<td>$id</td>";
    echo "<td>$nom</td>";
    echo "<td>$prenom</td>";
    echo "<td>$date</td>";
    echo "<td>";
    echo "<a href=eleve_edite.php?id=$id>voir</a>";
    echo "</td>";
    echo "</tr>";
}

echo "</table>";

?>

<form action="admin.php" method="post">
  <select name="ordre">
      <option value="nom">classement par nom</option>
      <option value="prenom">classement par
      ✂ prénom</option>
      <option value="naissance">classement par
      ✂ date</option>
  </select>
     <input type="submit" value="trier">
</form>

<?php
mysql_close($liendb);
include("bas.inc.php");
?>
```



LIMIT

Il est possible de limiter le nombre de résultats retournés par un SELECT avec LIMIT. Ainsi, SELECT * FROM eleve LIMIT 10 retourne 10 élèves. La commande LIMIT peut être complétée par un WHERE et un ORDER BY, mais doit cependant être placée à la fin de la requête.

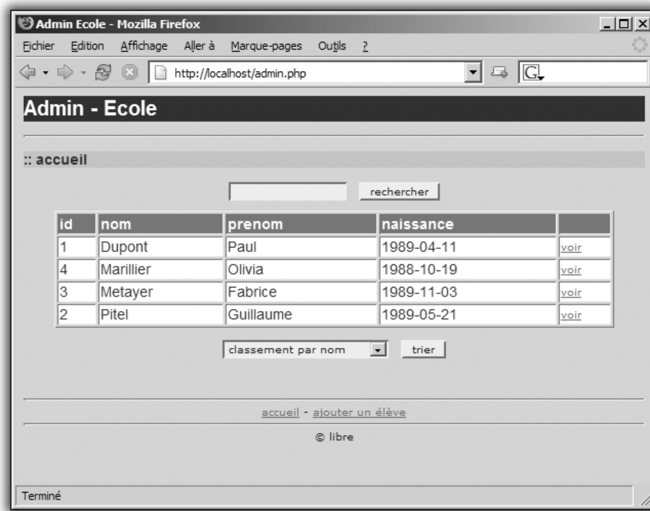


Figure 11.11 : *Classement par prénoms*

11.6. Check-list

- Une authentification HTML nécessite de travailler au niveau de l'en-tête HTTP et d'utiliser de la fonction `header()`.
- Les requêtes de type `UPDATE` permettent de mettre à jour des informations de la base.
- Les requêtes de type `DELETE` permettent de supprimer un enregistrement de la base.
- Il est toujours possible de limiter la portée d'une requête SQL en l'enrichissant d'une clause `WHERE`.
- La factorisation est une notion essentielle en informatique. Il est ainsi possible, pour un code PHP utilisé en différents endroits, de le placer dans un fichier extérieur et de l'inclure via la fonction `include()`.
- Au niveau de la présentation de la page, il est préférable de passer par un fichier CSS plutôt que placer des consignes visuelles au sein même des éléments qui la composent. L'évolution sera en effet largement facilitée.

Chapitre 12

La gestion des fichiers

Manipuler des fichiers 340
Créer des fichiers spéciaux 351
Check-list 382

Dans le cadre de ce chapitre, nous nous intéresserons principalement à la manipulation des fichiers en PHP. Vous étudierez dans un premier temps les différentes actions qui peuvent être réalisées sur un fichier, puis la présentation de différents formats de fichiers qui peuvent être générés avec PHP : GZIP (archives), CSV (Excel), PNG (image), PDF, SWF (Flash).

12.1. Manipuler des fichiers

Comme beaucoup de langages de programmation, PHP permet de manipuler les fichiers. Toutes les opérations habituelles sont disponibles :

- lire
- écrire
- renommer
- déplacer
- copier
- supprimer

Prenez un premier exemple : quand vous envoyez un fichier *hello.html* sur votre serveur d'hébergement (par FTP ou via le navigateur), une commande PHP vous permet en une ligne de le renommer :

Listing 12-1 : Renomme le fichier hello.html en hello.htm

```
rename("hello.html","hello.htm");
```

Ces fonctions permettent donc de manipuler les fichiers distants (présents sur le serveur d'hébergement).

Les fichiers de cache

Ces fonctions rendent possible le stockage de données dans des fichiers. Cela va a priori à l'encontre de ce qui a été dit précédemment au sujet des bases de données, à savoir qu'il est souvent préférable d'utiliser une base de données pour stocker des informations.

Il faut cependant garder à l'esprit que les requêtes SQL utilisent beaucoup de ressources et qu'il est judicieux, pour accroître les performances d'un script, de les réduire le plus possible.

La mise en œuvre d'une base de données est en effet très lourde à gérer pour le système : la couche réseau, le système de fichiers, ainsi que le SGBD sont mis à contribution.



Impact des requêtes SQL

Pour ceux qui souhaitent optimiser leur code, sachez que la diminution du nombre de requêtes SQL est de loin ce qui peut améliorer le plus les performances de vos scripts. Les optimisations sur les boucles et les diverses finesses du langage sont, en comparaison, négligeables.

Pour alléger les ressources du serveur et accélérer vos scripts, il peut donc être intéressant de mettre en place un système de cache.

Le principe d'un fichier de cache consiste à contenir de manière statique et pendant une durée déterminée des informations qui ne sont pas susceptibles d'être modifiées régulièrement, mais qui ont de fortes chances d'être réclamées fréquemment.

Pour créer ce fichier, il suffit d'extraire les informations de la base de données et de les écrire dans un fichier selon une certaine norme. L'avantage réside dans le fait que ce fichier devient aussi facile à traiter pour le serveur qu'un simple fichier *.html*. Dès lors, vous n'avez pas à craindre que l'afflux soudain d'un grand nombre d'internautes freine l'ensemble du site.

Imaginez qu'un site web partenaire ait besoin d'accéder aux noms et aux prénoms des élèves de l'école. Ce site n'a pas accès à la base de données et ne peut donc pas extraire directement les informations dont il a besoin. Vous allez par conséquent mettre à sa disposition un fichier de cache contenant la liste des noms et des prénoms. Ce fichier sera disponible à l'adresse **http://localhost/eleves.cache**. Votre politique de cache sera alors de mettre en œuvre deux scripts :

- Le script *ecrit-cache.php* va permettre d'écrire les informations dans le fichier.
- Le script *lit-cache.php*, sera utilisé par le site partenaire pour accéder à votre fichier via le Web (en sachant évidemment que votre partenaire dispose du PHP).

L'écriture

Commencez par le stockage de vos données dans le fichier *eleves.cache* :

Listing 12-2 : Le script ecrit-cache.php

```
<?php

$fichier = fopen ("eleves.cache", "w+");
$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");
$sql = "SELECT * FROM eleve";
$resultat = mysql_query ($sql);
while ($eleve = mysql_fetch_array ($resultat)) {
  $nom = $eleve['nom'];
  $prenom = $eleve['prenom'];
  fwrite($fichier, "$nom,$prenom\n");
}
mysql_close($liendb);
fclose($fichier);
echo "fichier enregistré";

?>
```

Comme vous pouvez vous en rendre compte, le principe de fonctionnement est assez similaire à celui des bases de données.

Il convient, dans un premier temps, de créer un « descripteur de fichier » avec `fopen()` : `$fichier`.

Le premier argument de `fopen()` correspond au nom du fichier avec lequel vous allez travailler. Dans cet exemple, le nom est simple : `eleves.cache`. Celui-ci peut cependant être moins évident. Il peut ainsi être précédé d'un chemin (*path*). Si vous écrivez `../eleves.cache`, cela signifie que vous ouvrez le fichier *eleves.cache* dans le répertoire au-dessus du répertoire courant.



Répertoire courant

Le répertoire courant est le répertoire où se trouve le script PHP qui a été appelé. En écrivant `eleves.cache`, vous ouvrez donc un fichier qui sera situé dans le même répertoire que le script d'appel.

Le nom du fichier peut aussi être une URL : il est ainsi possible d'ouvrir un fichier présent sur le Web (`http://localhost/eleves.cache`) ou sur un serveur FTP (`ftp://ftp.kernix/eleves.cache`).

Le deuxième argument transmis à `fopen()` correspond au mode d'ouverture du fichier. Plusieurs modes existent :

Tableau 12.1 : *Les modes d'ouverture d'un fichier avec fopen()*

Mode d'ouverture	Signification
`r`	Ouverture en lecture seule (*read*)
`w`	Ouverture en écriture seule (*write*) ; le fichier est créé s'il n'existe pas
`r+`	Ouverture en lecture/écriture
`w+`	Ouverture en lecture/écriture ; le fichier est créé s'il n'existe pas et, s'il existe, il est préalablement mis à zéro
`a`	Ouverture en écriture (*append*) ; le fichier est créé s'il n'existe pas et, s'il existe, le pointeur est placé à la fin du fichier
`a+`	Ouverture en lecture/écriture ; le fichier est créé s'il n'existe pas et, s'il existe, le pointeur est placé à la fin du fichier

Dans ce cas, choisissez le mode `"w+"` car vous devez écrire dans le fichier *eleves.cache*.

Lors du premier appel au script *ecrit-cache.php*, le fichier est créé ; lors des appels suivants, vous n'ajouterez pas de données aux fichiers (`"a+"`), mais remplacerez le contenu par de nouvelles données.

Une fois le fichier ouvert, il est possible d'y écrire avec la fonction `fwrite()`. Cette fonction prend, elle aussi, deux arguments : le premier argument est le descripteur de fichier et le second correspond à la chaîne de caractères que vous souhaitez stocker dans le fichier. Il est important de placer un retour chariot (`\n`) à la fin de la chaîne si l'on ne veut pas avoir toutes ces données sur la même ligne.

Il convient de noter qu'une manipulation de fichier se conclut toujours par la fermeture du fichier avec `fclose()`.

La lecture

Passons maintenant à la lecture du fichier . Nous supposons dans un premier temps que le script *lit-cache.php* est sur le même compte que le script *ecrit-cache.php*.

La lecture locale

Vous pouvez ouvrir le fichier en lecture (`"r"`) en l'appelant directement par son nom :

Listing 12-3 : Le script lit-cache.php

```
$fichier = fopen ("eleves.cache", "r");
$eleves = fread ($fichier, filesize("eleves.cache"));
fclose ($fichier);
echo $eleves;
```

Vous faites dans ce cas appel à la fonction `fread()`. Cette fonction prend deux arguments : le premier est le descripteur de fichier et le second correspond à la quantité de données (en octets) que vous souhaitez lire dans le fichier. Faites appel à la fonction `filesize()` pour trouver cette valeur. Cette fonction retourne la taille du fichier, dont le nom lui est passé en paramètre. Grâce à cette fonction, vous lisez l'intégralité du fichier avec un seul appel à `fread()`.

La lecture distante

Supposons maintenant que ce script soit hébergé sur un autre serveur. Le fichier doit cette fois être ouvert via le Web. Passez à `fread()` le nom `"http://localhost/eleves.cache"`.



Ouverture de fichier web

Il est impossible d'ouvrir des fichiers web en écriture car cela reviendrait à permettre à n'importe qui de modifier le contenu de vos pages !

À la différence de la version locale, vous ne pouvez pas utiliser la fonction `filesize()` avec un fichier distant. Vous êtes donc obligé de lire le fichier morceau par morceau avec une boucle `while` :

```
$eleves = "";
while ($str = fread($fichier,16))
{
  $eleves .= $str;
}
```

À chaque itération de la boucle `while`, vous complétez la variable `$eleves` avec 16 octets du fichier *eleves.cache*. C'est le descripteur de fichier qui garde en mémoire l'endroit où vous vous trouvez dans le fichier. La boucle s'arrête naturellement dès qu'il n'y a plus rien à lire dans le fichier.

Si vous souhaitez revenir au début d'un fichier, vous avez (au moins) deux possibilités : la première, peu élégante, consiste à fermer le fichier et à le rouvrir ; la seconde consiste à utiliser la fonction `rewind()`.

`rewind()` prend en paramètre un descripteur de fichier. Cette deuxième solution est beaucoup plus intéressante car moins consommatrice en ressources.

Écrivez maintenant le script *lit-cache.php* qui va lire le fichier *eleves.cache* via le Web :

Listing 12-4 : ouverture d'un fichier web

```
<?php

$fichier = fopen ("http://localhost/eleves.cache", "r");
$eleves = "";
while ($str = fread($fichier,16))
{
  $eleves .= $str;
}
fclose($fichier);
echo $eleves;

?>
```

Vous prenez conscience, avec cet exemple, de l'extrême simplicité du langage PHP, même pour réaliser des tâches a priori complexes sur les fichiers. Voyez la fonction `fread()` avec un autre exemple :

```
<?php

$url  = "http://www.google.com";

$fichier = fopen ($url, "r");
while ($str = fread ($fichier, 16))
{
```

```
  $src .= $str;
}
fclose($fichier);

echo "<textarea rows=10 cols=80>$src</textarea>";

?>
```

Ce script récupère les sources de la page `http://www.google.com` et les affiche dans une zone de texte.

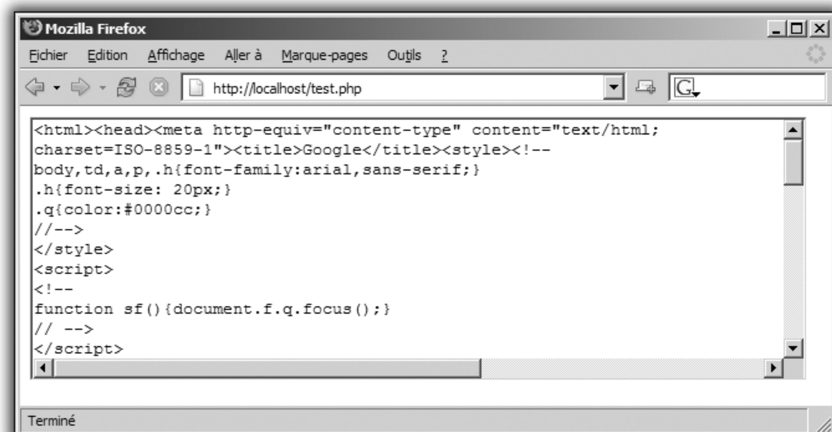


Figure 12.1 : *Affichage des sources de www.google.com*

Il est bien évidemment possible d'ouvrir plusieurs fichiers au même moment. Il suffit pour cela d'utiliser des noms de descripteurs différents.

Écrivez un petit script qui lit le fichier *eleves.cache* (via le Web) et qui génère un fichier *eleves2.cache*, où, cette fois, le nom apparaît après le prénom, avec un deux-points comme caractère de séparation. Utilisez les fonctions `file_get_contents()` (qui retourne le contenu d'un fichier) et `strlen()` pour trouver la taille du fichier distant.

Listing 12-5 : Le script lit-cache.php

```
<?php

$fichier1 = fopen ("http://localhost/eleves.cache", "r");
$fichier2 = fopen ("eleves2.cache", "w+");
$taille = strlen (file_get_contents("http://localhost
✂ /eleves.cache"));
$eleves = fread ($fichier1,$taille);
$tab_eleves = split ("\n",$eleves);
```

```
foreach ($tab_eleves as $ligne) {
  list ($nom,$prenom) = split(",",$ligne);
  fwrite($fichier2,"$prenom:$nom");
}
fclose ($fichier2);
fclose ($fichier1);

?>
```

Ne laissez pas traîner ce fichier *eleves2.cache*. La fonction `unlink()` permet de l'effacer :

```
unlink("eleves2.cache");
```

Vous vous apercevez, avec cette dernière fonction (ainsi qu'avec la fonction `rename()` vue plus haut), qu'il est inutile d'utiliser `fopen()` quand vous ne cherchez pas à travailler sur le contenu du fichier.

Les fichiers modèles : templates

Étudions enfin un dernier exemple qui illustre la lecture de fichier avec PHP. Vous avez vu dans les chapitres précédents l'importance, en programmation web, de la séparation entre le contenu et le visuel. L'utilisation de fichiers templates (modèles) est une bonne façon d'y parvenir. Le principe est très simple : plutôt que de mettre dans le même fichier le code PHP et la page HTML, stockez séparément ces deux parties. Le fichier *.html* comporte des balises indiquant ainsi où les informations issues de la base doivent être placées. Le script PHP se charge quant à lui de lire le fichier et de compléter les manques avec les données qu'il est allé récupérer dans la base (ou n'importe où ailleurs).

Dans cet exemple, le fichier *modele1.html* comporte deux balises qui seront remplacées :

- `%%NBELEVES%%` correspond au nombre d'élèves.
- `%%TABELEVES%%` sera remplacé par la liste des élèves.

Listing 12-6 : Le contenu du fichier modèle modele1.html

```
<html>
<body bgcolor="white">

<p>%%NBELEVES%% élèves</p>

<hr/>
```

```
<table align="center" border="1">
 <tr>
  <td>
   <font size="small">%%LISTELEVES%%</font>
  </td>
 </tr>
</table>

<hr/>

</body>
</html>
```

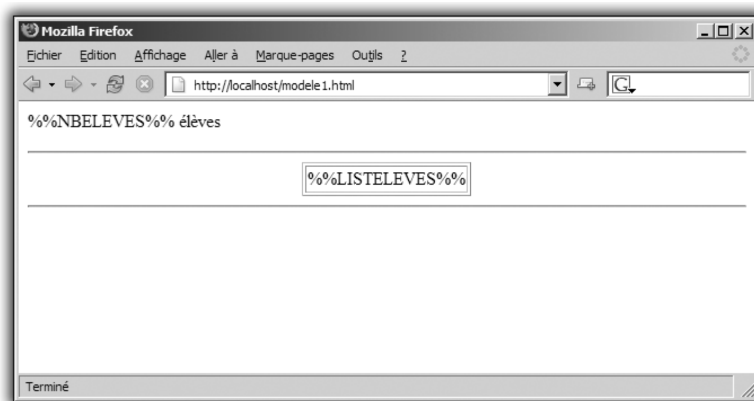


Figure 12.2 : *Le premier modèle*

Écrivez maintenant ce que l'on qualifie généralement de *wrapper*, c'est-à-dire le script qui va construire la page finale à partir du modèle :

```
<?php

$fichier = fopen ("modele1.html", "r");
$html = fread ($fichier, filesize("modele1.html"));
fclose ($fichier);

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");
$sql = "SELECT * FROM eleve";
$resultat = mysql_query ($sql);
$nb = mysql_num_rows($resultat);
$eleves = "";
while ($eleve = mysql_fetch_array ($resultat)) {
  $nom = $eleve['nom'];
  $prenom = $eleve['prenom'];
  $eleves .= "$nom,$prenom<br>";
}
```

```
mysql_close ($liendb);

$html = str_replace ("%%NBELEVES%%","$nb",$html);
$html = str_replace ("%%LISTELEVES%%",$eleves,$html);

echo $html;

?>
```

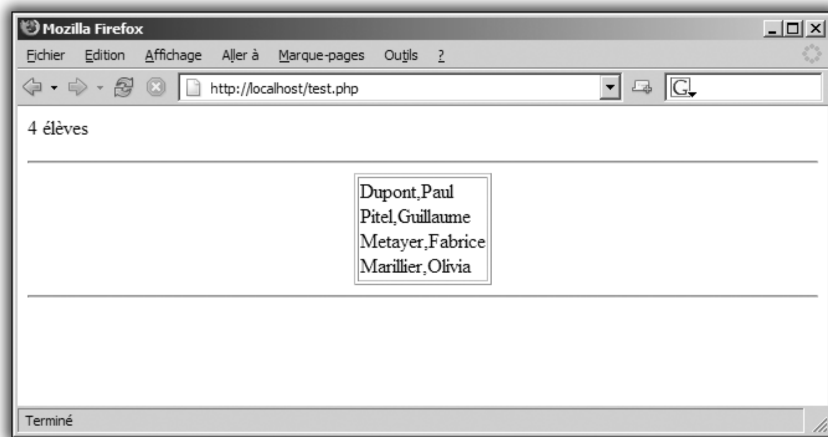


Figure 12.3 : *Les balises ont été remplacées*

Le grand avantage de ce mode de fonctionnement est de permettre de changer l'aspect visuel sans modifier le code. Cela rend la mise à jour de la page extrêmement aisée. Si vous travaillez avec un client, celui-ci peut réaliser un nouveau design très simplement sous Dreamweaver et vous transmettre le fichier en tant que modèle.

Imaginez maintenant que plusieurs partenaires veuillent pointer sur cette page, mais que chacun désire que la page soit habillée à sa façon. Rien de plus simple : vous prévoyez dans le script que le partenaire transmet en paramètre l'adresse du template, puis le script ouvre ce template via le Web pour générer la page.

Écrivez un deuxième template :

Listing 12-7 : autre modèle

```
<html>
<body bgcolor="#C7D0D9">

<table align="center" width="100%">
<tr><td bgcolor="white">%%NBELEVES%%</td></tr>
```

```
<tr><td bgcolor="#CCCCCC">%%LISTELEVES%%</td></tr>
</table>

</body>
</html>
```

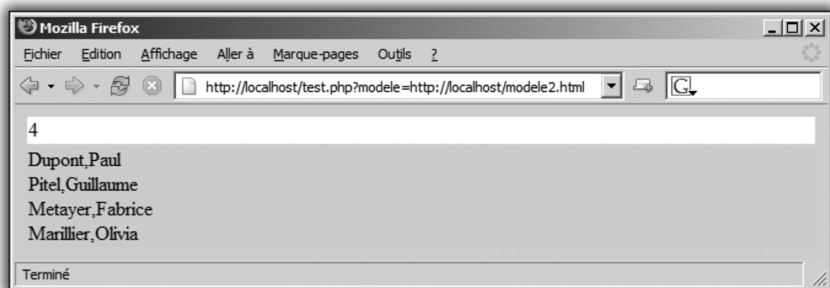


Figure 12.4 : *Le deuxième modèle*

Modifiez maintenant le script pour qu'il accepte en paramètre l'adresse du fichier modèle :

Listing 12-8 : le nom du modèle est transmis en paramètre

```
<?php

$fichier = fopen ($_REQUEST['modele'], "r");
while ($str = fread ($fichier, 16)) {
  $html .= $str;
}
fclose ($fichier);

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");
$sql = "SELECT * FROM eleve";
$resultat = mysql_query ($sql);
$nb = mysql_num_rows($resultat);
$eleves = "";
while ($eleve = mysql_fetch_array ($resultat)) {
  $nom = $eleve['nom'];
  $prenom = $eleve['prenom'];
  $eleves .= "$nom,$prenom<br>";
}
mysql_close ($liendb);

$html = str_replace ("%%NBELEVES%%","$nb",$html);
$html = str_replace ("%%LISTELEVES%%",$eleves,$html);

echo $html;
```

```
?>
```

Il devient alors possible de spécifier et d'héberger son propre modèle.

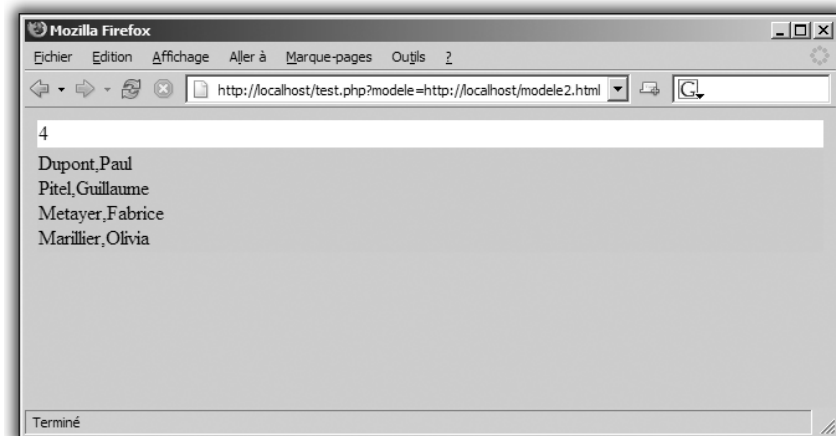


Figure 12.5 : *Page obtenue en tapant l'URL http://localhost/test.php?modele=http://localhost/modele2.html*

Cette solution sera de plus en plus courante car les applicatifs en ligne vont devenir peu à peu des services et il sera important que les clients de ces services puissent les adapter.

12.2. Créer des fichiers spéciaux

Vous avez vu qu'il était très simple, en PHP, de manipuler des fichiers. Vous allez voir, dans cette partie, qu'il est possible, aussi simplement, de générer des fichiers spéciaux : fichiers compressés, fichiers exploitables sous Excel, fichiers image.

Les fichiers compressés

Le PHP dispose de fonctions permettant de compresser des données. En les compressant, ces données prennent moins de place et sont donc beaucoup plus rapides à transférer.

Le format de compression par défaut en PHP est GZIP. Ce format est peu connu en dehors du monde Unix, mais il dispose de nombreux avantages :

- L'algorithme de compression est excellent.
- Des outils de décompression existent sur tous les systèmes d'exploitation (Stuffit Expander sous Mac, WinZip sous Windows).

Ce premier exemple aura pour objet de récupérer les noms et les prénoms des élèves, d'en faire une liste et de retourner la liste compressée afin de la stocker sur votre disque :

```
<?php
$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");
$sql = "SELECT * FROM eleve";
$resultat = mysql_query ($sql);
$liste = "";
while ($eleve = mysql_fetch_array ($resultat))
{
  $nom = $eleve['nom'];
  $prenom = $eleve['prenom'];
  $liste .= "$nom,$prenom\n";
}
mysql_close($liendb);
echo gzencode($liste);
?>
```

En appelant le fichier directement, vous obtenez un contenu illisible.

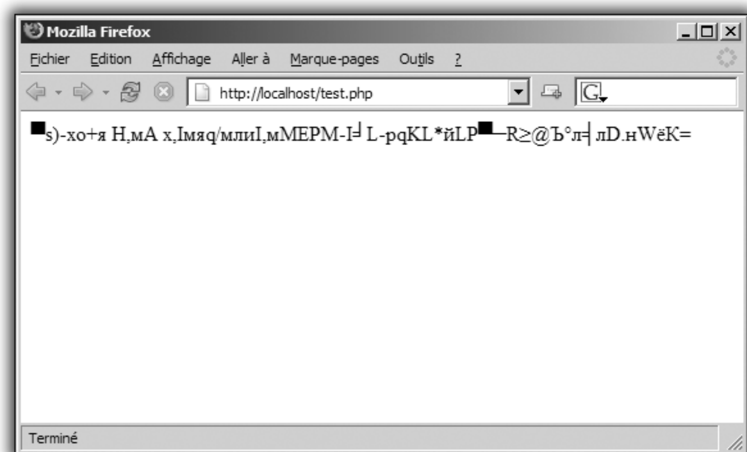


Figure 12.6 : *Contenu compressé illisible*

Le problème vient du fait que le fichier que vous appelez est considéré par le navigateur comme un simple fichier *.html*. Son contenu brut est

donc affiché. Il est cependant possible de donner des directives au navigateur pour lui indiquer que le contenu qu'il reçoit est compressé et qu'il doit par conséquent le sauvegarder plutôt que l'afficher.

La commande utilisée pour y parvenir est `header()`. Cette fonction permet de modifier l'en-tête d'un fichier transmis à un navigateur. Avec cette fonction, il est possible d'intervenir sur :

- les directives de cache (durée de conservation d'un fichier sur le disque) ;
- les dates de création ;
- le type de contenu du fichier.

Le rôle de la fonction `header()` se limite en fait à passer des commandes HTTP. La commande permettant de préciser le contenu du fichier envoyé est `Content-type`. Par défaut, le contenu est de type `text/html`. Si vous souhaitez indiquer que le fichier est compressé au format GZIP, il suffit d'écrire :

```
header("Content-type: application/x-gzip");
```

Il faut aussi ajouter une autre directive afin d'indiquer quel nom devra prendre le fichier à sauvegarder :

```
header('Content-Disposition:  attachement;
✂ filename="eleves.txt.gz"');
```



En-tête HTTP et courriel

La modification de l'en-tête HTTP d'un fichier est à rapprocher des ajouts que vous aviez faits dans l'en-tête d'un courriel pour préciser qu'il était en HTML. L'avantage de PHP est de permettre d'accéder à ces zones cachées et ainsi d'aller plus loin dans le code et les fonctionnalités. Il est par contre nécessaire de connaître les différents protocoles (HTTP, SMTP) pour pouvoir en tirer véritablement parti.

Complétez votre script avec ces deux nouvelles lignes :

```
<?php
header("Content-type: application/x-gzip");
header('Content-Disposition:  attachement;
✂ filename="eleves.txt.gz"');
$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");
$sql = "SELECT * FROM eleve";
$resultat = mysql_query ($sql);
```

```
$liste = "";
while ($eleve = mysql_fetch_array ($resultat))
{
  $nom = $eleve['nom'];
  $prenom = $eleve['prenom'];
  $liste .= "$nom,$prenom\r\n";
}
mysql_close($liendb);
echo gzencode($liste);
?>
```

Désormais, vous obtenez bien le comportement désiré. Le navigateur vous propose de sauvegarder le fichier sur le disque.

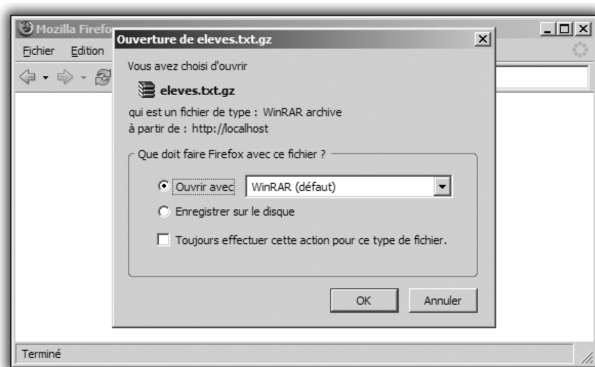


Figure 12.7 : *Le fichier compressé peut être sauvé sur le disque*

Une fois que le fichier se trouve sur le disque, il est possible de le décompresser :

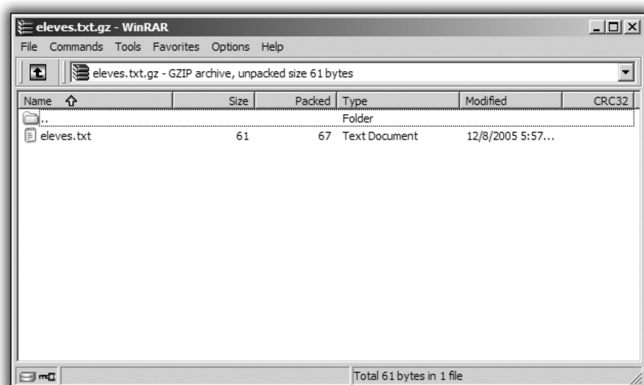


Figure 12.8 : *Ouverture du fichier compressé avec WinRAR*



Le format BZIP2

Un autre format de compression est disponible : BZIP2. L'avantage de ce format est d'offrir un meilleur taux de compression que le GZIP. Du fait de la jeunesse de ce format, les décompresseurs de fichiers GZIP ne sont pas très répandus. Cela interdit donc son usage si l'on souhaite conserver une certaine forme de compatibilité avec le plus grand nombre d'internautes.

Grâce à la commande `header()`, vous avez pu forcer le navigateur à sauver les données qui lui ont été transmises. Cette technique peut se révéler très intéressante si vous souhaitez associer un événement au téléchargement d'un fichier.

Plutôt que donner un lien fixe sur le fichier (tel `http://localhost/fichier.exe`), vous avez la possibilité de passer par un script qui pourra par exemple, dans un premier temps, vérifier un mot de passe, puis logger le téléchargement dans une base avant de transmettre les données au navigateur.



La fonction header()

La commande `header()` doit être utilisée avant que toute donnée soit affichée. Si vous laissez une simple ligne vide avant l'ouverture `<?php`, vous obtiendrez une erreur. En effet, le caractère de saut de ligne aura été affiché avant la commande `header()`.

L'exemple suivant retourne le fichier *prog.exe* présent sur le serveur si l'internaute s'est identifié avec le login `"test"` :

```
<?php

if ($login == "test")
{
  header("Content-type: application/octetstream");
  header('Content-Disposition:  attachement;
  ✂ filename="prog.exe"');
  // possibilité ici d'enregistrer l'événement dans une
  ✂ base de données
  $fichier = fopen ("prog.exe", "r");
  while ($str = fread ($fichier, 1024))
  {
     echo $str;
  }
  fclose($fichier);
```

```
  return(1);
}

?>

<form>
<input name='login' /><input type='submit'
✂ value='telecharger' />
</form>
```

Les fichiers Excel

Il est très courant, lorsque vous développez un applicatif en ligne, d'avoir besoin de générer un fichier d'export des données stockées dans votre base. Un tel fichier peut être utilisé par un logiciel tiers pour réaliser certaines opérations spécifiques. Vous pourriez ainsi imaginer d'exporter les courriels de tous les élèves et utiliser ce fichier avec un logiciel de publipostage afin d'envoyer un bulletin d'information à l'ensemble de la classe.

Il s'avère que la quasi-totalité du marché de l'informatique grand public est occupée par Windows et qu'Excel est l'outil le plus utilisé pour traiter des listings de données. Vous allez donc présenter, dans cette partie, une manière de générer des fichiers compatibles avec Excel.

La méthode se révèle relativement simple : il suffit de générer un fichier CSV (Comma Separated Value), c'est-à-dire un fichier où les enregistrements sont organisés ligne par ligne et où les données sont séparées par des virgules :

- `"Dupont","Paul"`
- `"Pitel","Guillaume"`
- `"Metayer","Fabrice"`
- `"Marillier","Olivia"`

Grâce à la fonction `header()` et à la directive `Content-type`, vous êtes en mesure d'ouvrir vos données directement sous Excel sans avoir besoin de passer par un fichier temporaire :

```
<?php
header("Content-type: text/x-csv");
header('Content-Disposition:  attachement;
✂ filename="list.csv"');
$liendb = mysql_connect("localhost", "root", "");
```

```
mysql_select_db ("test");
$sql = "SELECT * FROM eleve";
$resultat = mysql_query ($sql);
$liste = "";
while ($eleve = mysql_fetch_array ($resultat))
{
  $nom = $eleve['nom'];
  $prenom = $eleve['prenom'];
  $liste .= "\"$nom\",\"$prenom\"\r\n";
}
mysql_close($liendb);
echo $liste;
?>
```

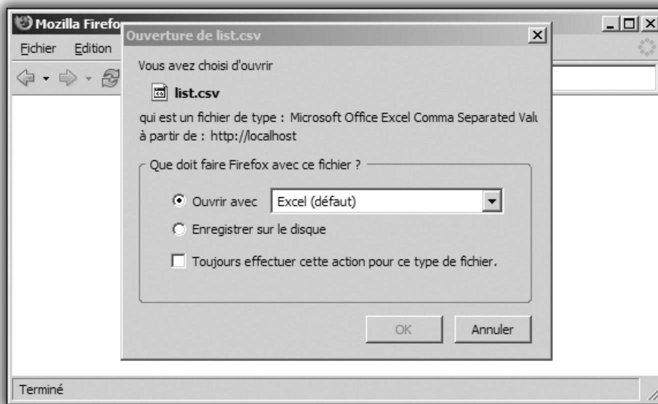


Figure 12.9 : *Le navigateur vous propose bien d'ouvrir le fichier avec Excel*

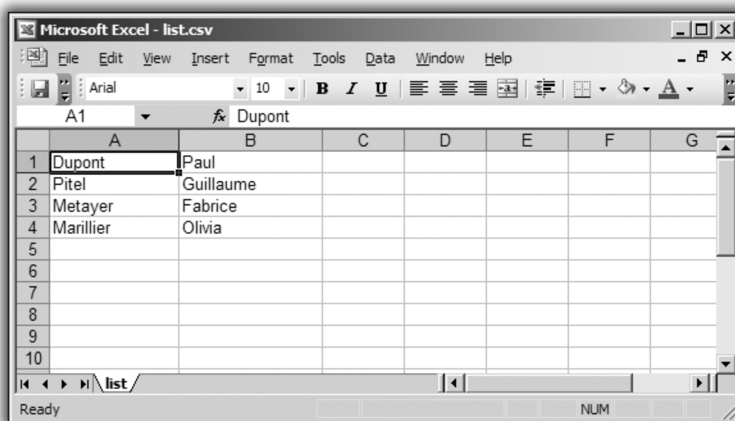


Figure 12.10 : *Les données sont directement lisibles dans Excel*

Dans Excel, il est possible de traiter vos données de manière très précise, d'utiliser des filtres, d'exécuter des opérations complexes. Il est souvent préférable de fonctionner avec une exportation de fichier vers un logiciel évolué plutôt que de recoder laborieusement une multitude de fonctionnalités qui ne serviront peut-être jamais. Il vaut mieux limiter l'applicatif en ligne à sa fonction première. Lorsque vous développez une boutique en ligne, il est sans doute préférable d'exporter les commandes vers un logiciel de comptabilité plutôt que de créer un module comptable.

Les fichiers Flash

PHP est un logiciel développé par des développeurs et pour des développeurs. Dans cette communauté, la notion de défi technique et de jeu prend souvent le pas sur les considérations purement économiques. Très rapidement, des modules plus ou moins farfelus sont ainsi venus enrichir la librairie d'extensions du PHP. Parmi ceux-ci, on peut trouver un module permettant de générer des fichiers SWF. Un fichier SWF est normalement généré avec le logiciel de Macromedia Flash. Avec PHP, il est cependant possible de créer dynamiquement des animations Flash.

Parmi tous les projets consacrés à ce sujet sur Internet, le projet Ming semble être le plus actif et le plus abouti. Grâce à cette extension, vous êtes en mesure, en quelques lignes de PHP, de réaliser de véritables animations qui peuvent interagir avec la souris, le clavier, être accompagnées de musique, etc.

Bien que Wamp Server soit livré par défaut avec l'extension permettant de générer des SWF, il ne l'autorise pas par défaut. L'étape préalable à la manipulation des SWF consiste donc à faire en sorte que PHP charge cette extension au démarrage. La première solution consiste à modifier le fichier *php.ini* en supprimant le point-virgule devant l'instruction suivante :

```
extension=php_ming.dll;
```

Le serveur Apache doit ensuite être redémarré pour prendre en compte cette modification.

La seconde solution consiste à passer par l'icône de Wamp Server et le menu **PHP extensions** pour sélectionner `php_ming`.



Extensions sous Linux

L'ajout d'extensions sous Linux se révèle beaucoup plus complexe. Vous êtes en effet obligé de récupérer les sources de l'extension ainsi que celles de PHP. Les deux devront être recompilées pour pouvoir disposer de cette nouvelle extension. Cette situation est essentiellement due au fait qu'il existe une multitude de distributions Linux et que ces dernières fonctionnent le plus souvent sur une grande variété de plateformes (PC, Mac, etc.). Cette diversité rend ainsi la distribution en mode binaire extrêmement fastidieuse pour les développeurs.

Le code suivant permet ainsi de jouer le fichier *test.mp3* (placé dans le même répertoire que le script) :

```
<?php
$m = new SWFMovie();
$m->setRate(12.0);
$m->streamMp3(fopen("test.mp3", "r"));
$m->setFrames(141);
header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

Notez le `Content-type : application/x-shockwave-flash`. En six lignes, vous êtes en mesure d'ajouter une dimension musicale à votre site. De plus, il est avantageux d'utiliser Flash dans ce cas car il s'agit de la technologie la plus portable : Flash existe en effet pour tous les navigateurs et les systèmes d'exploitation.

Le code suivant permet, quant à lui, de faire tourner sur lui-même un carré rouge :

```
<?php
$s = new SWFShape();
$s->setRightFill($s->addFill(0xff, 0, 0));
$s->movePenTo(-50,-50);
$s->drawLineTo(50,-50);
$s->drawLineTo(50,50);
$s->drawLineTo(-50,50);
$s->drawLineTo(-50,-50);
$p = new SWFSprite();
$i = $p->add($s);
for($j=0; $j<17; ++$j) {
  $p->nextFrame();
  $i->rotate(5);
}
$p->nextFrame();
$m = new SWFMovie();
```

```
$i = $m->add($p);
$i->moveTo(160,120);
$i->setName("blah");
$m->setBackground(0xff, 0xff, 0xff);
$m->setDimension(320,240);
header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

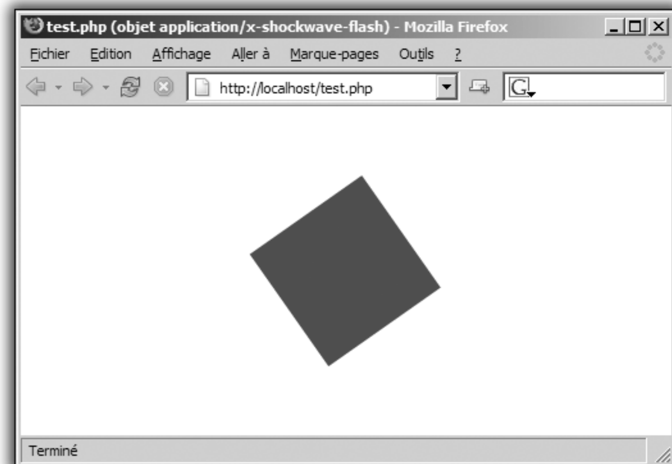


Figure 12.11 : *Flash généré par du code PHP*

Un des grands avantages de Flash est qu'il permet d'obtenir une interactivité en temps réel avec l'internaute. L'exemple suivant permet de déplacer un cercle vert sur un carré rouge et d'appeler la fonction Javascript extérieure `externAction()` lorsque le cercle est placé sur le carré :

Listing 12-9 : Script qui génère le FLASH

```
<?
ming_useswfversion(6);

$movie = new SWFMovie();
$movie->setRate(30.000000);
$movie->setDimension(500, 500);
$movie->setBackground(0xcc,0xcc,0xcc);

$square = new SWFShape();
$square->setLine(5,0,0,0);
$square->setRightFill(255,0,0);
$square->movePenTo(-75,-75);
$square->drawLine(150,0);
```

```
$square->drawLine(0,150);
$square->drawLine(-150,0);
$square->drawLine(0,-150);

$sprite = new SWFSprite();
$sprite ->add($square);
$sprite ->nextFrame();

$f = $movie->add($sprite);
$f->setName("square");
$f->moveTo(250,350);

$circle = new SWFShape();
$circle->setLine(5,0,0,0);
$circle->setRightFill(0,255,0);
$circle->drawCircle(50);

$sprite = new SWFSprite();
$sprite ->add($circle);
$sprite ->nextFrame();

$f = $movie->add($sprite);
$f->setName("circle");
$f->moveTo(250,100);

$movie->add(new SWFAction("
circle.onPress = function () {
  this.startDrag(1);
};
circle.onRelease = function () {
  stopDrag();
  if ((circle._x > (square._x - square._width/2)) &&
      (circle._x < (square._x + square._width/2)) &&
      (circle._y > (square._y - square._height/2)) &&
      (circle._y < (square._y + square._height/2))) {
    circle._x = square._x;
    circle._y = square._y;
    _root.getUrl('javascript:externAction()');
  }
};
"));

$movie->output();
?>
```

Listing 12-10 : page contenant la function javascript appelée et l'appel au FLASH

```
<html>
<head>
<title>Exemple Ming</title>
</head>
```

```
<script>
function externAction () {
  alert("Bravo !");
}
</script>
<body>
<p>Faire un drag&drop du cercle vert dans la carré rouge</p>
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/
✂ flash/swflash.cab#version=6,0,29,0" width="300" height="300">
  <param name="movie" value="test.php">
  <param name="quality" value="high">
  <param name="wmode" value="transparent">
  <embed src="test.php" width="300" height="300"
  ✂ wmode="transparent" quality="high"
  pluginspage="http://www.macromedia.com/go/getflashplayer"
  ✂ type="application/x-shockwave-flash"></embed>
</object>
</body>
</html>
```

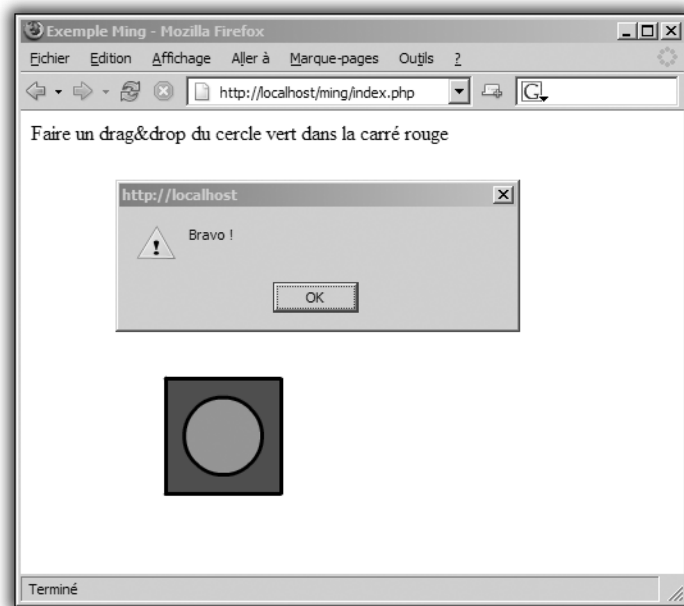


Figure 12.12 : *Déclenchement d’une action lorsque le cercle est placé sur le carré*

Si cette extension n’est pas présente chez votre hébergeur, n’hésitez pas à lui en parler : il y a de fortes chances pour qu’il vous l’installe

rapidement. Le projet Ming est hébergé à l'adresse **http://ming.sourceforge.net/**. Vous trouverez sur le site l'extension à télécharger ainsi qu'une documentation complète.

Les fichiers PDF

Le format de fichiers PDF est devenu depuis quelques années un standard pour les échanges de documents sur Internet. Le logiciel Acrobat Reader, de la société Adobe, est en effet présent sur toutes les plateformes (Windows, Mac, Unix). PHP donne la possibilité de créer à la volée des documents PDF. Cette technique peut être intéressante si vous souhaitez créer des documents téléchargeables à partir de données présentes dans votre base.

Comme pour le Flash, l'extension de gestion des PDF doit également être autorisée dans le fichier *php.ini*. La ligne à décommenter est cette fois : `extension=php_pdf.dll`. L'alternative consistant à sélectionner l'extension `php_pdf` dans le menu d'extensions de Wamp est également possible.

Écrivez maintenant un script qui va générer un document PDF proposant autant de pages que d'élèves, avec dans chaque page le nom et le prénom de l'élève :

Listing 12-11 : Script qui génère un document PDF

```
<?php

$pdf = pdf_new();

if (!pdf_open_file($pdf, "")) {
    print("error");
    exit(0);
};

pdf_set_info($pdf, "Author", "fx bois");
pdf_set_info($pdf, "Title", "exemple");
pdf_set_info($pdf, "Creator", "fx bois");
pdf_set_info($pdf, "Subject", "exemple");

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");
$sql = "SELECT * FROM eleve";
$resultat = mysql_query ($sql);
$liste = "";
while ($eleve = mysql_fetch_array ($resultat))
```

```
{
  $nom = $eleve['nom'];
  $prenom = $eleve['prenom'];
  pdf_begin_page($pdf, 595, 842);
  pdf_add_bookmark($pdf, "fiche $nom",0,0);
  $font = pdf_findfont($pdf, "Helvetica", "host", 0);
  if ($font) pdf_setfont($pdf, $font, 12);
  pdf_set_value($pdf, "textrendering", 1);
  pdf_show_xy($pdf, "$nom $prenom", 50, 750);
  pdf_moveto($pdf, 50, 740);
  pdf_stroke($pdf);
  pdf_end_page($pdf);
}

mysql_close($liendb);

pdf_close($pdf);

$buf = pdf_get_buffer($pdf);
$len = strlen($buf);

header("Content-type: application/pdf");
header("Content-Length: $len");
header("Content-Disposition: inline; filename=fichier
✂ -eleve.pdf");
print($buf);
pdf_delete($pdf);

?>
```

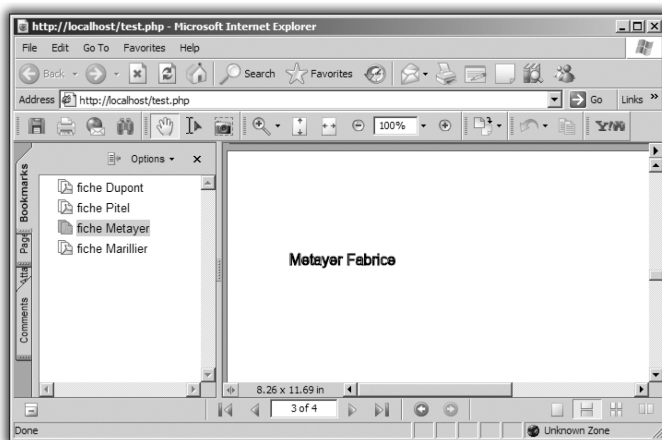


Figure 12.13 : *Document PDF généré par du PHP ouvert dans Internet Explorer*

Plutôt que de l'afficher directement, il est également possible de sauver le fichier PDF sur le disque en remplaçant les lignes suivantes :

```
header("Content-type: application/pdf");
header("Content-Length: $len");
header("Content-Disposition: inline; filename=fichier
✂ -eleve.pdf");
print($buf);
```

… par :

```
$fichier = fopen ("fiche-eleve.pdf", "w+");
fwrite($fichier, $buf);
fclose($fichier);
```

Les fichiers image

À la différence des extensions vues précédemment, le module de génération d'images est un des modules les plus populaires de PHP, et il y a de très fortes chances pour que vous puissiez en disposer chez votre hébergeur.

La directive à décommenter dans `php.ini` est : `extension=php_gd2.dll`.

Les fondamentaux

Ce module est basé sur la librairie GD qui contient tout le code permettant de réaliser les nombreuses manipulations graphiques. De nombreux autres langages de programmation tels que Perl ou Python permettent de l'interfacer.

Développée prioritairement pour les environnements Unix/Linux, cette librairie existe aussi sous Windows et peut donc permettre la génération d'image sur ce système.

Comme pour PDF ou Flash, vous utilisez des fonctions du module pour générer le contenu de l'image ainsi que la fonction `header()` pour faire comprendre au navigateur que le fichier est de type image.

Le module est en mesure de générer différents formats d'images, chacun disposant de ses points forts.

- JPEG : très bonne compression pour des images complexes (photos), très répandu.

- GIF : très bonne compression pour des images simples, très répandu, gère la transparence.
- PNG : très bonne compression générale, gestion avancée de la transparence.
- WBMP : il s'agit d'un format utilisé pour le WAP.

Aujourd'hui, le format le plus complet est le PNG. Son seul inconvénient est sa jeunesse. De ce fait, certains navigateurs anciens ne sont pas en mesure de le lire.



Format GIF

Les versions les plus récentes de la librairie GD autorisent de nouveau la création d'images au format GIF. L'algorithme de compression (LZW) est en effet tombé dans le domaine public il y a peu de temps.

Votre premier script va générer une image noire contenant un rectangle blanc :

Listing 12-12 : Première création d'image

```
<?php

header ("Content-type: image/png");
$image = @Imagecreate (150, 150);
$noir = ImageColorAllocate ($image, 0, 0, 0);
$blanc = ImageColorAllocate ($image, 255, 255, 255);
ImageFilledRectangle ($image, 10, 10, 20, 20, $blanc);
ImagePng ($image);

?>
```

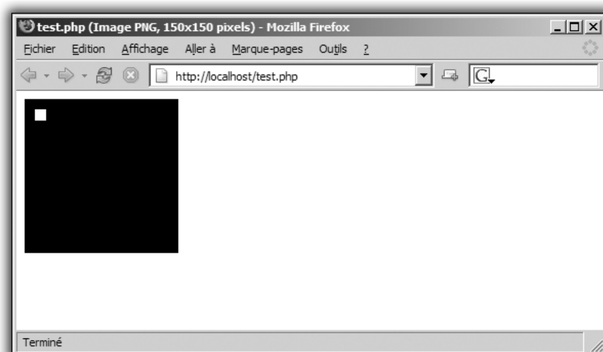


Figure 12.14 : *Génération d'une image noire contenant un rectangle blanc*

La création d'images est donc un processus très simple qui peut être divisé en plusieurs étapes :

1 Précisez que le fichier est de type image avec la commande `header()`. Dans ce cas, vous générez une image PNG en utilisant la directive `Content-type: image/png`. Si vous choisissez de générer une image JPEG, vous écrirez `Content-type: image/jpeg`, et `Content-type: image/gif` dans le cas d'une image GIF.

2 Créez l'image avec la fonction `Imagecreate()`. Vous transmettez à cette fonction la largeur et la hauteur et elle retourne un identifiant d'image : `$image`.

3 Définissez les couleurs utilisées dans l'image avec la fonction `ImageColorAllocate()`. Le premier argument est l'image elle-même (`$image`), les trois derniers arguments sont les composantes RVB (rouge, vert, bleu) de la couleur. La couleur noire est représentée par le triplet (0, 0, 0), la couleur blanche par (255, 255, 255) et la couleur rouge par (255, 0, 0). Cette manière de coder les couleurs est la même qu'en HTML, à la différence qu'ici les valeurs sont décimales alors qu'en HTML les couleurs sont hexadécimales : le rouge s'écrit par exemple `#FF0000` en HTML. La manière la plus simple de trouver les codes des couleurs est d'utiliser les palettes des outils de création graphique comme Photoshop, Paint Shop Pro ou Gimp. Il faut savoir que la première couleur définie servira de couleur de fond pour l'image. Dans le cas présent, il s'agit de la couleur noire (`$noir`).

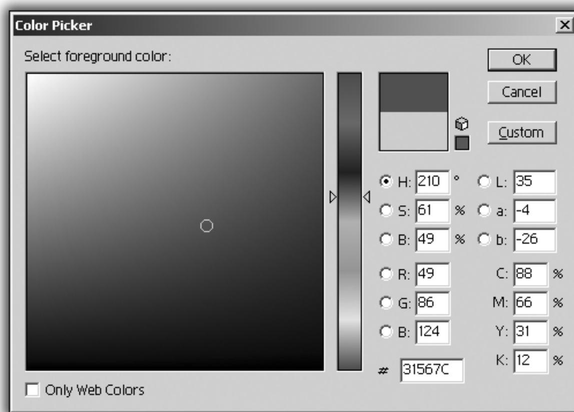


Figure 12.15 : *La palette de Photoshop donne le codage RVB (ou RGB : Red, Green, Blue)*

4 Composez l'image avec tous ses éléments : formes géométriques, textes, morceaux d'autres images. Dans ce script, choisissez simplement un carré blanc placé dans l'angle gauche de l'image. Utilisez, pour cela, la fonction `ImageFilledRectangle()` qui prend six arguments : l'identifiant de l'image, les deux coordonnées de l'angle supérieur gauche, les deux coordonnées de l'angle inférieur droit et enfin la couleur du rectangle.



Une liste complète des fonctions est fournie dans le chapitre consacré « Les fonctions PHP ».

5 La dernière étape consiste à générer le contenu de l'image avec la fonction `ImagePng()`. Les fonctions `ImageGIF`, `ImageJPEG`, `ImageWBMP` sont aussi disponibles pour les autres formats.

Dans l'exemple précédent, vous avez fait le choix d'appeler le script directement. Le navigateur n'affiche donc que l'image. Si vous souhaitez inclure l'image au sein d'une page web, il suffit de faire appel au script depuis la balise `<img>`. Écrivez la page HTML suivante dans ce sens :

Listing 12-13 : Inclusion d'image créée dynamiquement au sein d'une page HTML

```
<html>
<body>
<hr/>
<center>
 <img src="test.php" />
</center>
<hr/>
</body>
</html>
```

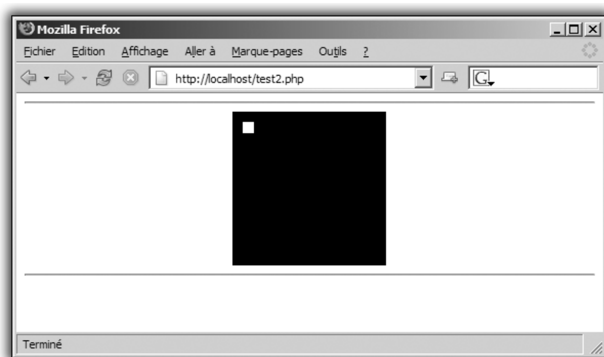


Figure 12.16 : *Image dans une page HTML*

Écrivez maintenant un deuxième exemple, où vous allez exploiter d’autres fonctionnalités de ce module (traçage de ligne, écriture) :

```
<?php

header ("Content-type: image/png");

$image = @Imagecreate (250, 200);

$couleur1 = ImageColorAllocate ($image, 212, 208, 200);
$couleur2 = ImageColorAllocate ($image, 198, 193, 182);
$couleur3 = ImageColorAllocate ($image, 79, 78, 74);


for ($i = 0; $i <= 200; $i += 10)
{
  ImageLine ($image, 0, $i, 250, $i, $couleur2);
}

ImageString ($image,5,10,20, "B o n j o u r",$couleur3);
ImageStringUp ($image,4,150,60, "Monde",$couleur3);

ImagePng ($image);

?>
```

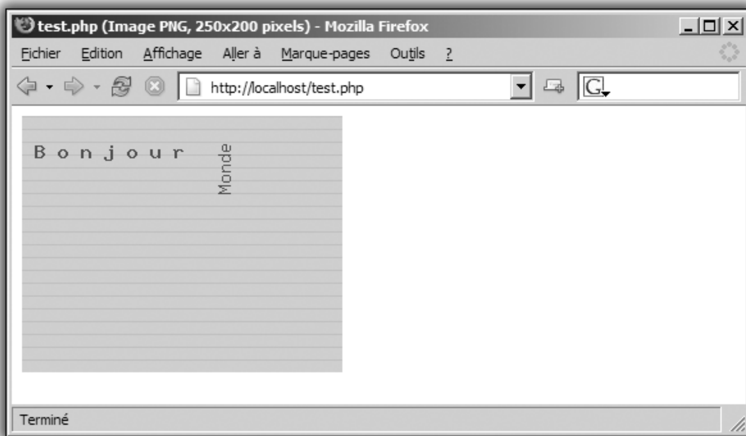


Figure 12.17 : *Autres fonctionnalités du module image*

Vous retrouvez dans cet exemple les principes vus plus haut.

La boucle `for()` permet de tracer des lignes tous les 10 pixels grâce à la fonction `ImageLine()`. Les arguments de cette fonction sont

l'identifiant de l'image, les coordonnées du point de départ, les coordonnées du point d'arrivée et la couleur.

Les textes sont, quant à eux, tracés avec les fonctions `ImageString()` et `ImageStringUp()` dont les arguments sont l'identifiant de l'image, la taille de la fonte, les coordonnées du point de départ, le texte et enfin la couleur. Le suffixe `Up` dans `ImageStringUp()` signifie que le texte est tracé verticalement.

Comme vous pouvez vous en apercevoir, la fonte est très sommaire. Heureusement, d'autres fonctions permettent de spécifier la fonte du texte : `ImageTTFText()`.



ImageTTFText sous Linux

Pour pouvoir utiliser cette fonction, PHP doit avoir été compilé avec le support de la librairie Freetype. Il n'est donc pas assuré que vous puissiez l'utiliser.

Cette fonction prend huit arguments : l'identifiant de l'image, la taille de la fonte, l'angle, les coordonnées de l'origine, la couleur, le nom de la fonte et le texte. Une fonte doit donc avoir été transférée sur le compte pour pouvoir l'utiliser. De nombreux sites web proposent le téléchargement gratuit de fontes : **www.fontfreak.com** ou **www.1001freefonts.com** en sont deux bons exemples.

```
<?php

header ("Content-type: image/png");

$image = @Imagecreate (250, 200);

$couleur1 = ImageColorAllocate ($image, 212, 208, 200);
$couleur2 = ImageColorAllocate ($image, 198, 193, 182);
$couleur3 = ImageColorAllocate ($image, 79, 78, 74);
$blanc    = ImageColorAllocate ($image, 255, 255, 255);

for ($i = 0; $i <= 200; $i += 10)
{
  ImageLine ($image, 0, $i, 250, $i, $couleur2);
}

ImageTTFText ($image,20,0,5,50,$couleur3,"font1.ttf","B o n
✂ j o u r");
ImageTTFText ($image,70,30,60,170,$blanc,"font2.ttf","Monde");
```

```
ImagePng ($image);

?>
```

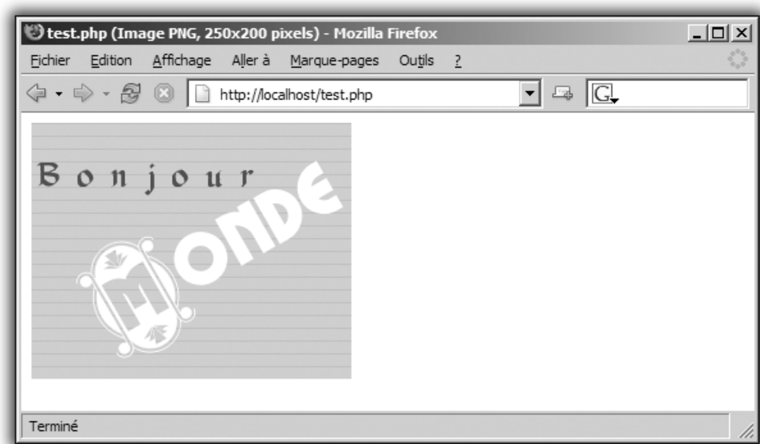


Figure 12.18 : *Des polices TrueType sont utilisées pour dessiner les textes*

Si la fonte se trouve dans un autre répertoire, par exemple *datas*, il faut alors écrire `datas/font1.ttf`.

Vous remarquez que l'écriture verticale est ici réalisée avec la même fonction que l'écriture horizontale. Vous vous contentez en fait de mettre un angle de 30 degrés.

PHP, en permettant d'accéder aux bases de données d'une part et de générer des images dynamiques d'autre part, est devenu un langage de choix pour la création d'outils de statistiques et de *reporting*.

Vous allez dans la suite de ce chapitre réaliser des graphiques reflétant les notes de vos élèves. Pour cela, il nous faut ajouter la notion de notes et de classement à votre base de données.

Les bases de données relationnelles

Comme vous l'avez vu précédemment, une base de données peut contenir plusieurs tables. L'avantage des BDD relationnelles telles que MySQL ou PostgreSQL est de permettre de créer des liens entre ces tables. On parle à propos de ces bases de SGBDR (système de gestion de bases de données relationnelles).

Supposons que vous vouliez enregistrer dans votre base de données des informations concernant l'élève, son professeur et ses notes. Spontanément, un utilisateur non expérimenté pourrait proposer d'organiser ses données dans une seule et grande table comportant les champs suivants : `nomeleve`, `prenomeleve`, `classement`, `moyenne`, `nomprofesseur`.

Cette manière de faire est à proscrire tout de suite !

Vous vous apercevez très vite, en remplissant la table, qu'il faut enregistrer le nom d'un même professeur pour tous les élèves de sa classe :

- `dupont, eric, 3, 16, mortier`
- `durand, michel, 12, 11, mortier`

Cela vous fait donc perdre à la fois du temps et de l'espace disque inutilement.

Un autre inconvénient de cette organisation est de ne pas permettre de conserver un historique des classements et des moyennes sur plusieurs trimestres.

Les BDD relationnelles viennent alors à votre secours. Si vous essayez de séparer intelligemment les données en plusieurs tables liées entre elles, ces inconvénients disparaissent instantanément. Vous pouvez en effet envisager l'organisation de vos données de cette manière : un professeur possède plusieurs élèves et un élève possède plusieurs moyennes et classements. En terme de BDD, on peut dire que la table des professeurs est liée à celle des élèves et que cette dernière est liée à celle des notes. Les liaisons correspondent en fait à la présence de l'identifiant (`idprof`) du professeur dans la table *eleve* et à l'identifiant (`ideleve`) de l'élève dans la table moyenne.

Vos trois tables sont donc :

- *prof* : `idprof`, `nom`, `prenom`
- *eleve* : `ideleve`, `idprof`, `nom`, `prenom`
- *exam* : `idexam`, `ideleve`, `moyenne`, `classement`, `trimestre`

Pour sélectionner le nom des élèves qui ont, d'une part, une moyenne supérieure à 10 au premier trimestre et qui, d'autre part, ont comme professeur M. Paul, vous pouvez utiliser la requête suivante :

```
SELECT eleve.nom AS nomeleve, prof.nom AS nomprof, exam
✂ .classement FROM prof, eleve, exam WHERE prof.nom =
✂ 'Paul' AND eleve.idprof = prof.idprof AND moyenne
✂ .ideleve = eleve.ideleve AND exam.classement >= '10' AND
✂ exam.trimestre = '1'
```

Analysons minutieusement cette requête. Comme vous avez besoin des trois tables, vous écrivez `FROM prof, eleve, exam`.

Les tables *prof* et *eleve* contiennent des colonnes intitulées de la même manière : *nom* et *prenom*. Il devient donc impossible, en travaillant sur ces deux tables en même temps, d'écrire `WHERE nom = 'dupont'` car le serveur de BDD ne peut savoir s'il doit faire une vérification sur la table *prof* ou sur la table *eleve*. Pour préciser de quel nom il s'agit, utilisez la notation `table.colonne`, dès que vous voulez spécifier une colonne dans une table en particulier :

```
WHERE prof.nom = 'Paul'
```

De la même manière, un obstacle apparaît si vous voulez sélectionner à la fois le nom de l'élève et le nom du professeur dans la même requête. La précédente méthode ne servirait ici à rien :

```
SELECT eleve.nom, prof.nom, exam.classement
```

Vous n'avez en effet aucun moyen, dans le résultat de la requête, de préciser si vous voulez le nom du professeur ou celui de l'élève. La technique consiste donc à renommer les colonnes au niveau de `SELECT : eleve.nom AS nomeleve`. De cette manière, en PHP, `$tab['nomeleve']` contient le nom de l'élève et `$tab['nomprof']` celui du professeur.



SELECT dans plusieurs tables

Il est donc très dangereux d'utiliser `SELECT table1.*, table2.*`, si *table1* et *table2* contiennent des colonnes de même intitulé.

Dans cet exemple, contentez-vous de créer une table *exam* :

```
CREATE TABLE exam (
  idexam int(10) unsigned NOT NULL auto_increment,
  ideleve int(10) unsigned NOT NULL default '0',
  moyenne float(5,2) unsigned NOT NULL default '0.0',
  trimestre tinyint(3) unsigned NOT NULL default '0',
  PRIMARY KEY  (idexam),
  KEY ideleve (ideleve)
)
```

Quelques remarques sur la création de cette table :

- La colonne des moyennes présente des nombres à virgule et est donc de type `FLOAT`. Il est possible avec les `FLOAT` de déterminer la précision. `(5,2)` signifie que le nombre peut s'étendre sur 5 caractères, avec 2 caractères après la virgule (`19,25` s'étend bien sur 5 caractères virgule comprise). Il est pertinent de bien fixer la précision d'un nombre à virgule directement dans MySQL, cela vous évite de faire des conversions en PHP.
- Vous avez placé un index sur la colonne *ideleve* car il y a de fortes chances pour que les requêtes sur cette table fassent intervenir précisément cette colonne.
- La présence du champ `ideleve` lie la table *eleve* à la table *exam* et crée une relation entre ces deux tables.

Ajoutez la variable `$table_exam` à votre fichier *variables.inc.php* et réalisez un petit script qui devra :

- créer la table ;
- associer une note par trimestre à chaque élève de la table.

Listing 12-14 : Création et initialisation de la table exam

```
<?php

include("variables.inc.php");

$liendb = mysql_connect($bddserver, $bddlogin,
✂ $bddpassword);
mysql_select_db ($bdd);

$sql = "CREATE TABLE $table_exam (
  idexam int(10) unsigned NOT NULL auto_increment,
  ideleve int(10) unsigned NOT NULL default '0',
  moyenne float(5,2) unsigned NOT NULL default '0.0',
  trimestre tinyint(3) unsigned NOT NULL default '0',
  PRIMARY KEY  (idexam),
  KEY ideleve (ideleve)
)";
mysql_query ($sql);

echo "création de la table effectuée<hr>";

$sql = "SELECT ideleve FROM $table_eleve";
$resultat = mysql_query ($sql);
$i = 0;
while ($eleve = mysql_fetch_array ($resultat))
```

```
{
  $tab_eleves[$i] = $eleve['ideleve'];
  $i++;
}

$i = 0;
srand ((double) microtime() * 1000000);
while ($ideleve = $tab_eleves[$i])
{
  echo "eleve [$ideleve] :";
  for ($j = 1; $j <= 3; $j++)
 {
    $moyenne = rand(0,20);
    echo " $moyenne";
    $sql = "INSERT INTO $table_exam
    ✂ (ideleve,moyenne,trimestre) VALUES
    ✂ ($ideleve,$moyenne,$j)";
    mysql_query ($sql);
 }
 echo "<br>";
 $i++;
}

echo "<hr>moyennes enregistrées<hr>";

mysql_close($liendb);

?>
```



Figure 12.19 : *Initialisation de la table et des moyennes*

Créer des graphiques

Vous allez, dans cette partie, réaliser un script qui permettra de voir l'évolution de la moyenne de la classe sur les trois trimestres. Le graphique va être du type `bar chart`. Suivez pas à pas la façon de le construire.

L'image est de type PNG :

```
header ("Content-type: image/png");
```

Elle a pour dimensions 340 x 200 :

```
$image = @Imagecreate (340, 220);
```

La couleur de fond est le même vert que vous avez utilisé dans le back-office :

```
$fond = ImageColorAllocate ($image, 208, 216, 213);
```

D'autres couleurs sont utilisées pour les axes, les légendes, etc. Déclarez-les dès maintenant :

```
$coul_axes = ImageColorAllocate ($image, 11, 62, 43);
$coul_lignes = ImageColorAllocate ($image, 227, 235, 232);
$coul_legendes = ImageColorAllocate ($image, 11, 62, 43);
$coul_barres = ImageColorAllocate ($image, 42, 124, 94);
$coul_orange = ImageColorAllocate ($image, 207, 140, 53);
```

Deux axes, vertical et horizontal, sont présents :

```
imageline ($image,30,30,30,190,$coul_axes);
imageline ($image,30,190,320,190,$coul_axes);
```

Terminez le script avec la génération de l'image :

```
ImagePng ($image);
```

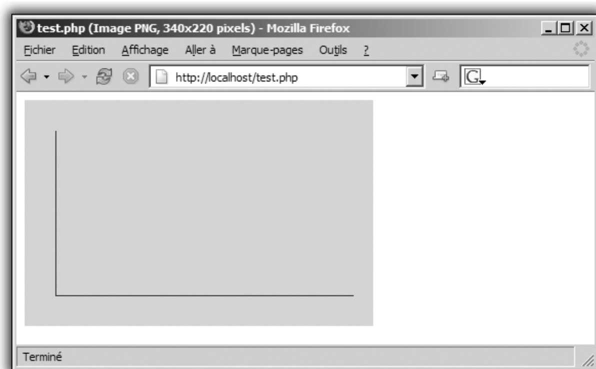


Figure 12.20 : *Génération d'un graphique*



Origine d'une image

L'angle supérieur gauche de l'image a pour coordonnées (0,0). Les coordonnées vont donc de gauche à droite et de haut en bas.

Dessinez les flèches à l'extrémité des axes. Pour cela, vous devez utiliser la fonction `imagefilledpolygon()` qui permet de dessiner un polygone (une forme géométrique à n côtés). Cette fonction prend quatre arguments :

- l'identifiant ;
- un tableau contenant les coordonnées des différents points formant le polygone (x0, y0, x1, y1, etc.) ;
- le nombre de sommets du polygone ;
- la couleur du polygone.

```
$tab_fleche_ord = array (30, 30, 26, 34, 34, 34);
$tab_fleche_abs = array (320, 190, 316, 186, 316, 194);

imagefilledpolygon ($image, $tab_fleche_ord, 3, $coul_axes);
imagefilledpolygon ($image, $tab_fleche_abs, 3, $coul_axes);
```

Pour vous rendre compte de l'avancée, définissez des affichages intermédiaires. Le code contient pour l'instant les instructions suivantes :

```
<?php

header ("Content-type: image/png");

// création de l'image
$image = @Imagecreate (340, 220);

// définition des couleurs
$fond       = ImageColorAllocate ($image, 208, 216, 213);
$coul_axes = ImageColorAllocate ($image, 11, 62, 43);
$coul_lignes = ImageColorAllocate ($image, 227, 235, 232);
$coul_legendes = ImageColorAllocate ($image, 11, 62, 43);
$coul_barres = ImageColorAllocate ($image, 42, 124, 94);
$coul_orange = ImageColorAllocate ($image, 207, 140, 53);

// les axes
imageline ($image,30,30,30,190,$coul_axes);
imageline ($image,30,190,320,190,$coul_axes);

// les flèches au bout des axes
$tab_fleche_ord = array (30, 30, 26, 34, 34, 34);
```

```
$tab_fleche_abs = array (320, 190, 316, 186, 316, 194);

imagefilledpolygon ($image, $tab_fleche_ord, 3, $coul_axes);
imagefilledpolygon ($image, $tab_fleche_abs, 3, $coul_axes);

// génération de l'image
ImagePng ($image);

?>
```

Passez maintenant à l'affichage des légendes des axes :

```
ImageTTFText
✂ ($image,10,0,5,20,$coul_legendes,"arial.ttf","moyenne");
ImageTTFText ($image,10,0,280,180,$coul_legendes,"arial
✂ .ttf", "trimestre");
```

L'axe des ordonnées dispose d'une graduation de 0 à 20 :

```
imageline ($image,26,190,30,190,$coul_axes);
imageline ($image,26,155,30,155,$coul_axes);
imageline ($image,26,120,30,120,$coul_axes);
imageline ($image,26,85,30,85,$coul_axes);
imageline ($image,26,50,30,50,$coul_axes);
```

Les ordonnées sont indiquées au niveau de la graduation :

```
ImageTTFText ($image,8,0,6,190,$coul_legendes,"arial.ttf", "0");
ImageTTFText ($image,8,0,6,155,$coul_legendes,"arial.ttf", "5");
ImageTTFText ($image,8,0,6,120,$coul_legendes,"arial.ttf", "10");
ImageTTFText ($image,8,0,6,85,$coul_legendes,"arial.ttf", "15");
ImageTTFText ($image,8,0,6,50,$coul_legendes,"arial.ttf", "20");
```

Pour faciliter la lecture des valeurs, affichez une trame horizontale légère :

```
imageline ($image,31,155,320,155,$coul_lignes);
imageline ($image,31,120,320,120,$coul_lignes);
imageline ($image,31,85,320,85,$coul_lignes);
imageline ($image,31,50,320,50,$coul_lignes);
```

Affichez maintenant les barres avec des valeurs fixes.

Moyenne de la classe :

- Premier trimestre : 13,5.
- Deuxième trimestre : 12.
- Troisième trimestre : 16.

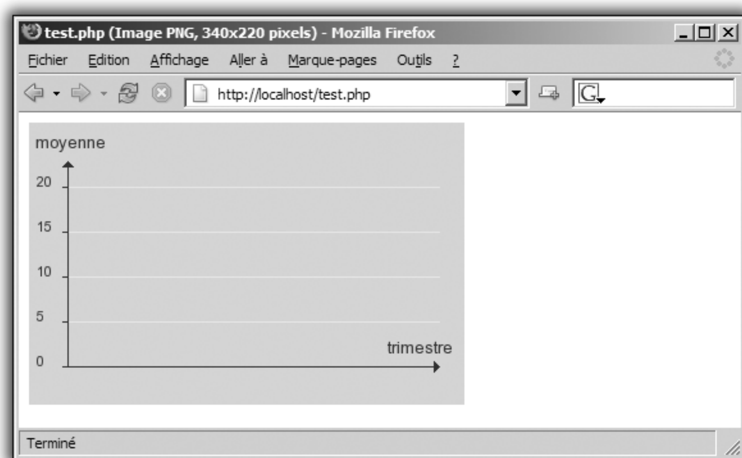


Figure 12.22 : *Ajout de la graduation et des légendes*

La difficulté est de trouver la hauteur des barres. Vous disposez de 140 pixels (190-50) pour afficher des données qui vont au maximum jusqu'à 20. Comme les coordonnées vont de haut en bas, c'est en fait la différence entre la note et la note maximale (20) qui doit vous intéresser. Une simple règle de trois permet ensuite de trouver l'ordonnée du sommet des barres : `(20 – note) * 7 + 50`.

```
imagefilledrectangle ($image, 40, (20-13.5)*7+50, 110,
189, $coul_barres);
imagefilledrectangle ($image, 120, (20-12)*7+50, 190, 189,
$coul_barres);
imagefilledrectangle ($image, 200, (20-16)*7+50, 270, 189,
$coul_barres);
```

Affichez dans les barres la valeur de la moyenne :

```
ImageTTFText ($image,10,0,50,180,$coul_orange,"arial.ttf",
✂ "13.5");
ImageTTFText ($image,10,0,130,180,$coul_orange,"arial.ttf",
✂ "12");
ImageTTFText ($image,10,0,210,180,$coul_orange,"arial.ttf",
✂ "16");
```

Regroupez maintenant toutes les parties au sein d'un script *graph.php* et allez chercher les véritables valeurs dans la table *exam* :

Listing 12-15 : Script graph.php

```
<?php
include("variables.inc.php");
```

```
// récupération de la moyenne de la classe sur les 3 trimestres
$liendb = mysql_connect($bddserver, $bddlogin, $bddpassword);
mysql_select_db ($bdd);
$sql = "SELECT * FROM $table_exam";
$resultat = mysql_query ($sql);

// le tableau $tab contient 3 cases pour les 3 trimestres
// (de 0 à 2)
// chaque case contient la somme de toutes les moyennes
// des élèves pour le trimestre donné
$i = 0;
while ($tmp = mysql_fetch_array ($resultat))
{
  $tab[$tmp['trimestre'] - 1] += $tmp['moyenne'];
  $i++;
}
mysql_close($liendb);

// la variable $i contient le nombre de notes totales dans
// la table exam elle permet d'obtenir le nombre d'élèves
$nb_eleves = $i / 3;

// utilisez la fonction number_format pour n'avoir que
// 2 chiffres après la virgule
for ($i = 0; $i < 3; $i++)
{
  $tab[$i] = number_format($tab[$i] / $nb_eleves, 2);
}

//-----------------------------------------------------------
header ("Content-type: image/png");

// création de l'image
$image = @Imagecreate (340, 220);

// définition des couleurs
$fond      = ImageColorAllocate ($image, 208, 216, 213);
$coul_axes = ImageColorAllocate ($image, 11, 62, 43);
$coul_lignes = ImageColorAllocate ($image, 227, 235, 232);
$coul_legendes = ImageColorAllocate ($image, 11, 62, 43);
$coul_barres = ImageColorAllocate ($image, 42, 124, 94);
$coul_orange = ImageColorAllocate ($image, 207, 140, 53);

// les axes
imageline ($image,30,30,30,190,$coul_axes);
imageline ($image,30,190,320,190,$coul_axes);

// les flèches au bout des axes
$tab_fleche_ord = array (30, 30, 26, 34, 34, 34);
$tab_fleche_abs = array (320, 190, 316, 186, 316, 194);
imagefilledpolygon ($image, $tab_fleche_ord, 3, $coul_axes);
```

```
imagefilledpolygon ($image, $tab_fleche_abs, 3, $coul_axes);

// légendes
ImageTTFText ($image,10,0,5,20,$coul_legendes,"arial
✂ .ttf","moyenne");
ImageTTFText ($image,10,0,280,180,$coul_legendes,"arial
✂ .ttf","trimestre");

// graduations
imageline ($image,26,190,30,190,$coul_axes);
imageline ($image,26,155,30,155,$coul_axes);
imageline ($image,26,120,30,120,$coul_axes);
imageline ($image,26,85,30,85,$coul_axes);
imageline ($image,26,50,30,50,$coul_axes);

// ordonnées
ImageTTFText ($image,8,0,6,190,$coul_legendes,"arial .ttf","0");
ImageTTFText ($image,8,0,6,155,$coul_legendes,"arial .ttf","5");
ImageTTFText ($image,8,0,6,120,$coul_legendes,"arial .ttf","10");
ImageTTFText ($image,8,0,6,85,$coul_legendes,"arial .ttf","15");
ImageTTFText ($image,8,0,6,50,$coul_legendes,"arial .ttf","20");
// lignes légères
imageline ($image,31,155,320,155,$coul_lignes);
imageline ($image,31,120,320,120,$coul_lignes);
imageline ($image,31,85,320,85,$coul_lignes);
imageline ($image,31,50,320,50,$coul_lignes);

// affichage des barres
imagefilledrectangle ($image, 40, (20 - $tab[0]) * 7 + 50,
✂ 110, 189, $coul_barres);
imagefilledrectangle ($image, 120, (20 - $tab[1]) * 7 +
✂ 50, 190, 189, $coul_barres);
imagefilledrectangle ($image, 200, (20 - $tab[2]) * 7 +
✂ 50, 270, 189, $coul_barres);

// affichage de la valeur de la barre
ImageTTFText ($image,10,0,50,180,$coul_orange,"arial .ttf",$tab[0]);
ImageTTFText ($image,10,0,130,180,$coul_orange,"arial .ttf",$tab[1]);
ImageTTFText ($image,10,0,210,180,$coul_orange,"arial .ttf",$tab[2]);
ImagePng ($image);

?>
```

Ajoutez finalement la ligne suivante en bas du script *admin.php* afin d’avoir un aperçu de la moyenne de la classe :

```
<img src="graph.php" />
```

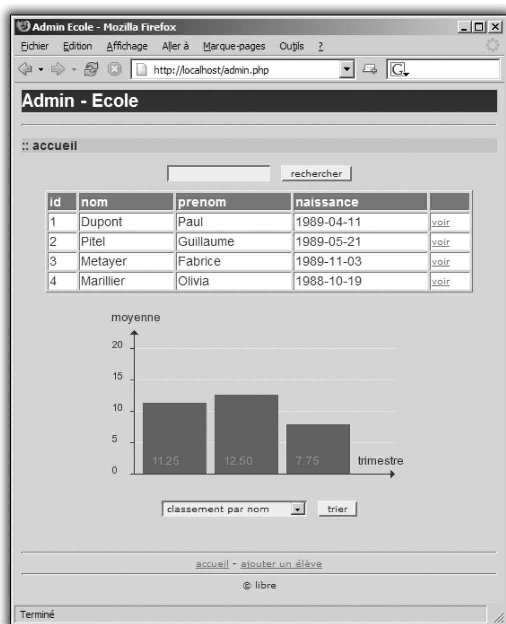


Figure 12.23 : *Un graphique inclus dans la page*

12.3. Check-list

- PHP dispose de toutes les fonctions permettant de travailler sur les fichiers texte : ouverture, modification, suppression, déplacement, etc.
- La technique dite des fichiers de cache est particulièrement intéressante pour alléger les ressources utilisées par un script.
- Des extensions peuvent être chargées afin de permettre à PHP de générer des fichiers plus complexes : PDF, Flash, image, etc.
- PHP, via la librairie GD, permet de travailler très finement sur les images. Cette extension est compatible avec la plupart des formats de fichiers image actuels : GIF, JPEG, PNG, etc.
- La génération d'un fichier spécial nécessite l'utilisation de la fonction `header()`.
- Le format XML permet de stocker des informations et se révèle être le meilleur choix pour les échanges de fichiers structurés.

Chapitre 13

La programmation objet

Classes et objets 385
Les méthodes magiques 393
Polymorphisme 398
Les interfaces 401
Itérateurs 403
Exceptions 405
Réflexion 409
Version objet de la génération de graphique 410
Check-list 416

Les exemples étudiés jusqu'à maintenant étaient développés de façon procédurale. Fonctions et variables étaient mélangées au sein de vos sources, sans réelle cohérence ni logique.

La programmation orientée objet (POO) propose un nouveau paradigme en vous permettant de manipuler des entités disposant chacune de leurs propres informations et traitements. L'applicatif final consiste ensuite à faire fonctionner ces différents objets de façon collaborative et intelligente.

Cette façon d'opérer facilite à la fois

- la modélisation ;
- la maintenance ;
- la relecture ;
- l'évolutivité.

Longtemps décriée, la dimension objet de PHP a gagné ses lettres de noblesse avec l'arrivée de la version 5 du langage. Cette évolution devenait indispensable dans une industrie informatique où tout applicatif d'envergure se doit d'être écrit en objet.

Parmi les évolutions majeures, la version 5 a notamment apporté :

- une distinction claire entre la copie et la duplication d'objets ;
- la gestion des interfaces ;
- l'opérateur `instanceof` ;
- le mot-clé `final` ;
- les constantes de classe ;
- les méthodes abstraites ;
- les méthodes et les attributs statiques ;
- les constructeurs et destructeurs ;
- la syntaxe `$monObjet->bonjour()->monde()` ;
- les exceptions ;
- les itérateurs ;
- la possibilité de définir une fonction `__autoload()` ;
- l'apparition de différents niveaux de visibilité : `public`, `protected`, `private`.

Plus généralement, les concepteurs du langage tendent à rendre la plateforme PHP réellement objet en fournissant la plupart des nouvelles extensions sous forme de classes (ex: PDO, XML, SOAP, etc.).

Cette « objectisation » correspond précisément au défi majeur que PHP devra relever dans les années à venir :

1. Devenir un véritable langage objet où tout élément est lui-même objet (types de données, I/O, etc.) afin d'attirer les développeurs chevronnés et les acteurs majeurs de l'industrie informatique (chasse gardée actuelle de Java et C#).
2. Ne pas perdre cette facilité d'accès inégalable qui lui a permis de devenir le premier langage de programmation web mondial.

13.1. Classes et objets

Deux notions fondamentales interviennent en POO : les classes et les objets.

Classes

Une classe peut être assimilée à un moule qui permet de créer des objets. Cette opération de création est appelée une instanciation.

Déclaration d'une classe

Le mot-clé `class` est utilisé pour définir une classe.

Listing 13-1 : Définition de la classe Rectangle

```
class Rectangle {

}
```

Cette classe doit maintenant être complétée de fonctions et de données qui lui sont propres. En POO, les fonctions sont appelées méthodes et les données, attributs. La classe Rectangle peut disposer par exemple :

- des méthodes : `surface()`, `perimetre()` ;
- des attributs : `longueur`, `largeur` et `couleur`.

Les méthodes `surface()` et `perimetre()` ont la possibilité d'accéder aux attributs de la classe par l'intermédiaire de la variable `$this` qui peut être assimilée à une référence à l'objet lui-même.

Listing 13-2 : Déclaration de la classe Rectangle

```
<?php

class Rectangle {

  public $longueur = null;
  public $largeur = null;
  public $couleur = "rouge";

  function perimetre() {
    if ($this->longueur!=null &&
        $this->largeur!=null) {
      return (2*$this->longueur+2*$this->largeur);
    }
  }

  function surface() {
    if ($this->longueur!=null &&
        $this->largeur!=null) {
      return ($this->longueur*$this->largeur);
    }
  }

}
```



Nom des méthodes

Il est vivement déconseillé de commencer le nom des méthodes par deux caractères __ (caractère espace souligné). PHP utilise en effet cette norme pour nommer ses méthodes, dites « magiques ».

Initialisation des attributs

Un attribut ne peut être initialisé qu'avec une valeur constante. Les initialisations suivantes ne sont par conséquent pas valides :

Listing 13-3 : Initialisation interdites

```
public $date = date();
public $id = "Paul"."Dupont";
```

La fonction `array()` peut cependant être utilisée si tous les éléments du tableau correspondent à des données statiques.

Listing 13-4 : Initialisation autorisée

```
public $tab = array("abc",123);
```

Objets

L'instanciation d'un objet utilise le mot-clé `new`. Un objet `$rectangle` peut être créé avec la syntaxe suivante :

Listing 13-5 : Instanciation d'un objet

```
$rect = new Rectangle();
```

Une fois l'objet créé, vous pouvez accéder à ses attributs et ses méthodes avec le séparateur –>.

Listing 13-6 : Récupération de la valeur d'un attribut et utilisation d'une méthode

```
$rect = new Rectangle();
echo $rect->couleur;

$rect->longueur = 3;
$rect->largeur = 5;
$tmp = $rect->perimetre();
```

PHP autorise également l'utilisation d'une variable pour spécifier un attribut derrière le séparateur –>.

Listing 13-7 : L'attribut est spécifié à l'aide d'une variable

```
$rect = new Rectangle(3,5);
$attr = "longueur";
$rect->$attr = 4;
```



Création dynamique d'attributs

PHP permet de créer des attributs à la volée :

```
$monrect = new Rectangle();

$monrect->toto = 1;
```

L'objet `$monrect` contient désormais l'attribut `$toto`.

La fonction `var_dump()` permet de visualiser les attributs d'une classe.

Listing 13-8 : Utilisation de var_dump()

```
$rect = new Rectangle();
print("<pre>");
var_dump($rect);
```

```
print("</pre>");
```

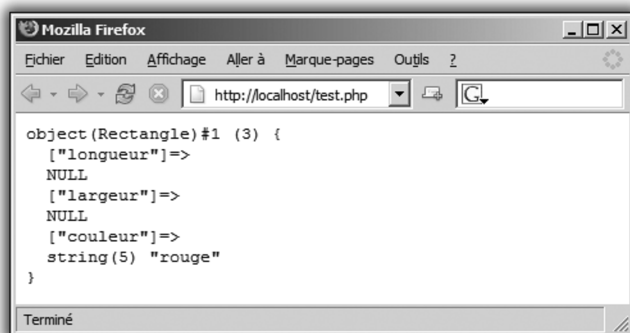


Figure 13.1 : *Contenu de l'objet $rect*

La vérification de la correspondance d'un objet à l'instanciation d'une classe donnée peut être réalisée avec l'opérateur `instanceof`.

Listing 13-9 : Vérification qu'un objet est bien l'instance d'une classe donnée

```
$rect = new Rectangle(3,5);

if ($rect instanceof Rectangle) {
  print("L'objet est un rectangle");
}
```



Variables globales et classes

Le mot-clé `global` et la « super globale » `$GLOBALS` peuvent être utilisés au sein d'une classe pour faire référence aux variables globales du script.

Fonction __autoload()

PHP doit disposer, pour instancier un objet, de la définition de la classe associée. Les sources de la classe peuvent être placées au sein même du script ou dans un fichier extérieur inclus via la fonction `require_once()`. Cette seconde possibilité a l'avantage d'autoriser le partage de la classe avec d'autres scripts de l'applicatif.

L'inclusion d'un fichier de définition de classe est moins fastidieuse depuis PHP5. Si votre script dispose d'une fonction `__autoload()`, PHP y fera appel pour chaque instanciation d'objet de la classe duquel il ne dispose pas.

Listing 13-10 : Définition de la fonction __autoload() qui ira chercher toutes les définitions de classe dans le répertoire class

```
function __autoload($nom_classe) {
  require_once("class/".$nom_classe.".php");
}
```



`__autoload()`

La fonction `__autoload()` ne doit être définie qu'une fois dans votre script !

Mot-clé static et constantes de classe

Les attributs et les méthodes dont les définitions sont complétées du mot-clé `static` peuvent être utilisés sans nécessiter l'instanciation d'un objet. Le séparateur `::` remplace alors –> pour y accéder.

Listing 13-11 : Utilisation d'un attribut et d'une méthode static

```
class Rectangle {
  public static $nom = 'rectangle';
  static function presentation() {
    print("Bonjour je suis un rectangle");
  }
}

echo Rectangle::$nom;
Rectangle::presentation();
```

Une méthode (`static` ou non) ne peut utiliser la variable `$this` pour accéder à un élément `static` (attribut ou méthode) de la classe. En effet, vous n'êtes pas au niveau objet mais au niveau classe. Le mot-clé `self` est alors utilisé pour faire référence à la classe elle-même.

Listing 13-12 : Utilisation d'un attribut static au sein de la classe

```
class Rectangle {
  public static $nom = 'rectangle';
  static function presentation() {
    print("Bonjour je suis un ".self::$nom);
  }
}
```

Les constantes de classe ne diffèrent des attributs `static` que du point de vue de leur syntaxe : le mot-clé `const` est utilisé pour leur déclaration et aucun `$` ne les précède.

Listing 13-13 : Utilisation d'une constante de classe

```
class Rectangle {
  const NOM = 'rectangle';
  static function presentation() {
    print("Bonjour je suis un ".self::NOM);
  }
}

echo Rectangle::NOM;
```

Comme les attributs, ces constantes ne peuvent être initialisées qu'avec des valeurs statiques.

Conversion

PHP donne la possibilité de convertir un objet en tableau et inversement. Le moyen pour y parvenir consiste à utiliser une opération de *cast*.

Listing 13-14 : Conversion d'un objet en tableau.

```
class Test {
  public $a = 1;
  public $b = 2;
  function hello {}
}
$test = new Test();
$tab = (array) $test;
```

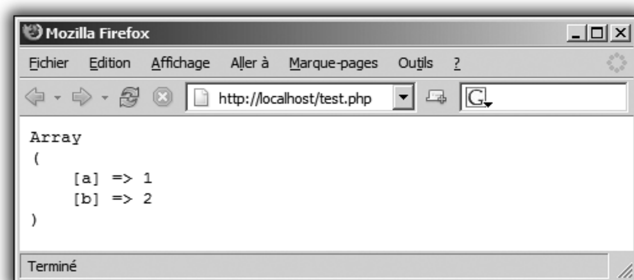


Figure 13.2 : *Tous les attributs sont conservés*

Listing 13-15 : Conversion d'un tableau en objet

```
$tab = array("a"=>1, "b"=>2);
$obj = (object) $tab;
var_dump($obj);
```

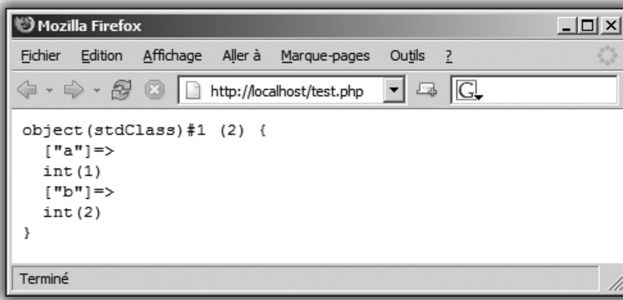


Figure 13.3 : *Conversion du tableau*

Constructeur et destructeur

La classe Rectangle reposant sur deux attributs essentiels (`$longueur` et `$largeur`), la grande majorité des instanciations sera suivie de leur initialisation. Pour éviter cette situation, la POO prévoit l'utilisation d'une fonction spécifique : le constructeur. En PHP, le constructeur correspond à une méthode portant le nom `__contruct()`. Cette fonction est appelée au moment de l'initialisation de l'objet et reçoit les arguments transmis à la classe lors de l'instanciation de l'objet. L'utilisation principale d'un constructeur reste l'initialisation des attributs de la classe.

Listing 13-16 : Initialisation automatique de l'objet grâce au constructeur

```
class Rectangle {

  public $longueur = null;
  public $largeur = null;
  public $couleur = "rouge";
  const NOM = "Super Rectangle";

  function __construct($longueur,$largeur) {
    $this->longueur = $longueur;
    $this->largeur = $largeur;
  }

  function perimetre() {
    if ($this->longueur!=null &&
        $this->largeur!=null) {
      return (2*$this->longueur+2*$this->largeur);
    }
  }
```

```
  function surface() {
    if ($this->longueur!=null &&
        $this->largeur!=null) {
      return ($this->longueur*$this->largeur);
    }
  }

}
$rect = new Rectangle (3,5);
```

PHP prévoit également la possibilité de définir une méthode `__destruct()` qui, si elle existe, sera appelée juste avant la suppression de l'objet. Ce destructeur est extrêmement utile pour les objets utilisant des ressources extérieures telles qu'une base de données, un fichier ou un service web. Au moment de la destruction de l'objet, le destructeur se charge alors de nettoyer son environnement en fermant les ressources ouvertes.

Listing 13-17 : Appels au constructeur et au destructeur

```
class Rectangle {

  public $longueur = null;
  public $largeur = null;
  public $couleur = "rouge";
  const NOM = "Super Rectangle";

  function __construct($longueur,$largeur) {
    print("Hello ".self::NOM."<br/>");
    $this->longueur = $longueur;
    $this->largeur = $largeur;
  }

  function perimetre() {
    if ($this->longueur!=null &&
        $this->largeur!=null) {
      return (2*$this->longueur+2*$this->largeur);
    }
  }

  function surface() {
    if ($this->longueur!=null &&
        $this->largeur!=null) {
      return ($this->longueur*$this->largeur);
    }
  }

  function __destruct() {
    print("Bye bye ".self::NOM."<br/>");
  }
```

```
}

$rect = new Rectangle(3,5);
unset($rect);
```

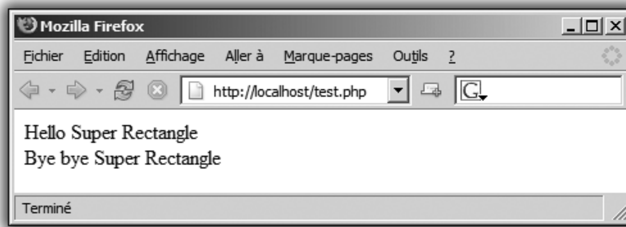


Figure 13.4 : *Les deux fonctions sont bien appelées*



Valeur `null`

Le destructeur est également appelé lorsque la variable `$rect` prend la valeur `null`.

13.2. Les méthodes magiques

Il s'agit de méthodes optionnelles normalisées par PHP et appelées suite à certains événements sur les objets. Les méthodes `__construct()` et `__destruct()` appelées lors de la création et la destruction d'un objet sont de bons exemples de méthodes magiques.

__sleep() et __wakeup()

Les fonctions `serialize()` et `unserialize()` présentées dans le chapitre consacré aux tableaux peuvent également être utilisées sur des objets. Dans un tel cas, PHP appelle, si elles existent dans la classe, les méthodes `__sleep()` avant la sérialisation et `__wakeup()` après la désérialisation.

Cette fonctionnalité est particulièrement utile lorsque votre objet repose sur une connexion ou un fichier extérieur. La fonction `__sleep()` peut alors être utilisée pour fermer les connections et `__wakeup()` pour les rouvrir.

__toString()

Cette méthode est appelée lorsque PHP se retrouve à devoir afficher un objet avec `print()` ou `echo()` (l'objet doit être seul entre parenthèses !). Si cette méthode existe, PHP affichera la valeur de retour de `__toString()` plutôt que le message `Object id #X`. Cette fonctionnalité est particulièrement utile pour debugger vos scripts.

Listing 13-18 : Utilisation de la méthode __toString()

```
class Rectangle {

  public $longueur = null;
  public $largeur = null;
  public $couleur = "rouge";
  const NOM = "Super Rectangle";

  function __construct($longueur,$largeur) {
    $this->longueur = $longueur;
    $this->largeur = $largeur;
  }

  function __toString() {
    return (self::NOM." [".$this->longueur."x".
                              $this->largeur."]");
  }
}

$rect = new Rectangle(3,5);
print($rect);
```

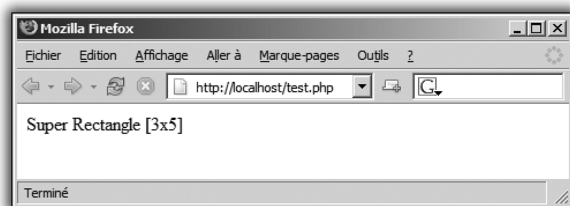


Figure 13.5 : *La fonction print()affiche la valeur retournée par la méthode __toString()*

Surcharge des accesseurs

Lorsque la méthode `__call()` est incluse au sein d'une classe, PHP l'utilise pour chaque appel à une méthode non définie de l'objet. Le

premier argument de `__call()` correspond au nom de la méthode appelée et le second argument, à un tableau des différents arguments de la méthode appelée.

Listing 13-19 : Utilisation de la méthode __call()

```
class Rectangle {

  public $longueur = null;
  public $largeur = null;
  public $couleur = "rouge";
  const NOM = "Super Rectangle";

  function __construct($longueur,$largeur) {
    $this->longueur = $longueur;
    $this->largeur = $largeur;
  }

  function __call($methode,$arguments) {
    return ("Appel à la methode ".$methode."() de
    ✂ ".self::NOM );
  }

  function surface() {
    if ($this->longueur!=null &&
        $this->largeur!=null) {
      return ($this->longueur*$this->largeur);
    }
  }

}

$rect = new Rectangle(3,5);
echo $rect->coucou()."<br/>";
echo $rect->surface();
```



Figure 13.6 : *L'appel de la méthode coucou()conduit à un appel de __call()*

Associée à la fonction `call_user_func()`, la méthode `__call()` peut être utilisée pour tracer puis déléguer de façon générique les appels à des méthodes internes.

Listing 13-20 : Utilisation de __call() pour déléguer les appels de méthodes

```
class Rectangle {
  public $longueur = null;
  public $largeur = null;
  public $couleur = "rouge";
  const NOM = "Super Rectangle";

  function __construct($longueur,$largeur) {
    $this->longueur = $longueur;
    $this->largeur = $largeur;
  }

  function __call($methode,$arguments) {
    print ("Méthode [ ".$methode." ]<br/>");
    $tmp = '_'.$methode;
    return call_user_func(array($this,$tmp),$arguments);
  }

  function _perimetre() {
    if ($this->longueur!=null &&
        $this->largeur!=null) {
      return (2*$this->longueur+2*$this->largeur);
    }
  }

  function _surface() {
    if ($this->longueur!=null &&
        $this->largeur!=null) {
      return ($this->longueur*$this->largeur);
    }
  }

}

$rect = new Rectangle(3,5);
echo $rect->surface();
echo "<br/>";
echo $rect->perimetre();
```

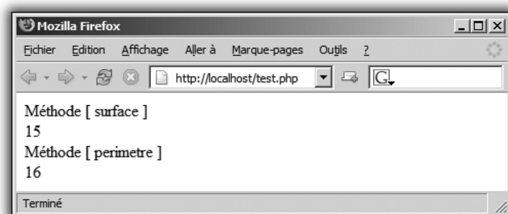


***Figure 13.7 :** Chaque appel à une méthode est tracé*

Les méthodes `__set()` et `__get()` sont appelées lorsque vous souhaitez modifier ou récupérer des attributs qui n'existent pas.

L'exemple suivant montre comment utiliser ces deux méthodes pour travailler avec l'attribut virtuel `dim`.

Listing 13-21 : Utilisation des méthodes __get() et __set()

```
class Rectangle {

  public $longueur = null;
  public $largeur = null;
  public $couleur = "rouge";
  const NOM = "Super Rectangle";

  function __construct($longueur,$largeur) {
    $this->longueur = $longueur;
    $this->largeur = $largeur;
  }

  function __set($nom,$val) {
    if ($nom=="dim" && count($val)==2) {
      list ($this->longueur, $this->largeur) = $val;
    }
  }

  function __get($nom) {
    if ($nom=="dim") {
      return array($this->longueur,$this->largeur);
    }
  }

  function surface() {
    if ($this->longueur!=null &&
        $this->largeur!=null) {
      return ($this->longueur*$this->largeur);
    }
  }

}

$rect = new Rectangle(3,5);
$rect->dim = array(2,3);
$dim = join("x",$rect->dim);
echo "Surface d'un rectangle de $dim = ";
echo $rect->surface();
```

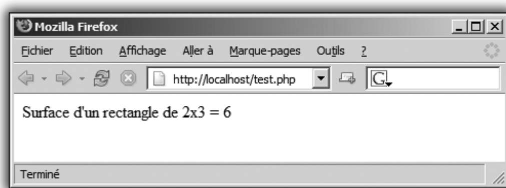


Figure 13.8 : *Affichage du résultat*

Sur le même principe, les méthodes `__isset()` et `__unset()` peuvent être définies.

13.3. Polymorphisme

Le polymorphisme est un des principes les plus importants de la programmation objet. L'idée principale consiste à créer un lien d'héritage entre deux classes.

Principe général

En dérivant d'une classe mère, la classe fille hérite de ses méthodes et de ses attributs qui viendront s'ajouter à celles et ceux déjà définis. Encore plus intéressant, la classe fille peut redéfinir les attributs et les méthodes de la classe mère : on dit alors qu'elle les surcharge. En favorisant l'héritage entre vos classes vous augmentez la généricité de vos classes et favorisez leur réutilisation. Tout le code générique est placé dans les classes parentes, et les codes spécifiques dans les classes filles.

Au niveau syntaxique, une classe B hérite de la classe A en utilisant le mot-clé `extends` :

Listing 13-22 : La classe B étend la classe A

```
class B extends A {

}
```

En considérant qu'un carré est un rectangle dont les côtés sont égaux, vous pouvez modéliser cette relation en déclarant une classe Carré qui étend la classe Rectangle.

Le mot-clé `parent` permet d'accéder aux méthodes et attributs de la classe mère. Il est particulièrement utile pour faire « remonter » les paramètres d'initialisation à la classe mère.

Listing 13-23 : La classe Carre hérite de la méthode surface() de la classe Rectangle

```
class Rectangle {

  public $longueur = null;
  public $largeur = null;
  const NOM = "Rectangle";
```

```
  function __construct($longueur,$largeur) {
    $this->longueur = $longueur;
    $this->largeur = $largeur;
  }

  function surface() {
    if ($this->longueur!=null &&
        $this->largeur!=null) {
      return ($this->longueur*$this->largeur);
    }
  }

}

class Carre extends Rectangle {

  public $cote = null;
  const NOM = "Carré";

  function __construct($cote) {
    parent::__construct($cote,$cote);
    $this->cote = $cote;
  }

}

$carre = new Carre(5);
echo $carre->surface();
```



Mot-clé `final`

Le mot-clé `final` accolé à la définition d'une classe interdit qu'elle soit étendue. Accolé à la définition d'une méthode, il interdit que cette méthode soit surchargée.

Visibilité

Une notion de visibilité peut être associée à la définition des méthodes et des attributs. La visibilité par défaut correspond à `public` et autorise un accès sans restriction. Les modes `private` et `protected` limitent quant à eux les accès de la façon suivante :

- `private` signifie que seule la classe associée peut accéder à la méthode (ou l'attribut) ;

- `protected` signifie que seules les classes disposant d'un lien d'héritage peuvent accéder à la méthode (ou l'attribut).

La visibilité doit précéder la définition de l'attribut ou de la méthode :

Listing 13-24 : Exemple de définition d'un attribut en mode protected et d'une méthode en mode private

```
protected $str = "";
private function maMethode() {}
```

PHP émet une erreur en cas de non-respect de la règle de visibilité :

Listing 13-25 : L'accés à une méthode protégée déclenche l'affichage d'une erreur

```
class Rectangle {

  private $longueur = null;
  private $largeur = null;

  function __construct($longueur,$largeur) {
    $this->longueur = $longueur;
    $this->largeur = $largeur;
  }

  protected function surface() {
    if ($this->longueur!=null &&
        $this->largeur!=null) {
      return ($this->longueur*$this->largeur);
    }
  }

}

$rect = new Rectangle(3,2);
echo $rect->surface();
```

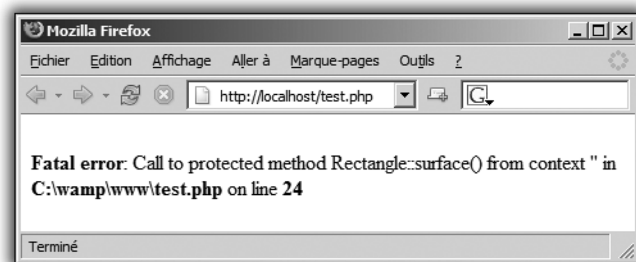


Figure 13.9 : Erreur correspondant à l'accès à une méthode protégée

13.4. Les interfaces

Certains langages de programmation permettent à une classe d'hériter de plusieurs classes mères. Les concepteurs de PHP n'ont pas souhaité mettre œuvre cet héritage multiple et ont préféré travailler sur la notion d'interface.

Une interface correspond à un ensemble de définitions de méthodes qu'une classe devra surcharger si elle souhaite l'implémenter. Une classe peut implémenter plusieurs interfaces.

La définition d'une interface utilise le mot-clé `interface` suivi d'un bloc contenant la définition des méthodes.

Listing 13-26 : Définition d'une interface

```
interface MonInterface {
  public function methode1 ();
  public function methode2 ();
  // etc.
}
```

Une classe implémente une interface en utilisant le mot-clé `implements`.

Listing 13-27 : Chaque classe doit implémenter la méthode aCotesEgaux()

```
interface ObjetGeom {
  public function aCotesEgaux ();
}

class Rectangle implements ObjetGeom{

  private $longueur = null;
  private $largeur = null;

  function __construct($longueur,$largeur) {
    $this->longueur = $longueur;
    $this->largeur = $largeur;
  }

  function aCotesEgaux () {
    if ($this->largeur==$this->longueur) {
      return true;
    }
    return false;
  }

}
```

```
class Carre extends Rectangle {

  public $cote = null;

  function __construct($cote) {
    parent::__construct($cote,$cote);
    $this->cote = $cote;
  }

  function aCotesEgaux () {
    return true;
  }

}
```

Toutes les méthodes des interfaces doivent être implémentées par la classe. Une erreur est déclenchée dans le cas contraire.

Listing 13-28 : La classe Carre n'implémente pas getNbCotes()

```
interface ObjetGeom {
  public function aCotesEgaux ();
  public function getNbCotes ();
}

class Carre implements ObjetGeom {

  public $cote = null;

  function __construct($cote) {
    parent::__construct($cote,$cote);
    $this->cote = $cote;
  }

  function aCotesEgaux () {
    return true;
  }

}
```

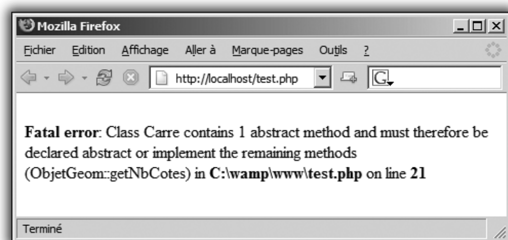


Figure 13.10 : *Affichage du résultat*

Une interface peut être assimilée à un contrat que doit respecter la classe qui l'implémente.

13.5. Itérateurs

La boucle `foreach()` peut être utilisée sur un objet afin d'afficher ses différents attributs.

```
$rect = new Rectangle(3,5);
foreach ($rect as $attr => $val) {
  print("$attr = $val<br/>");
}
```

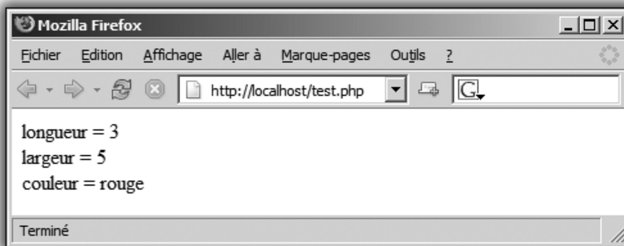


Figure 13.11 : *Affichage des différents attributs de l'objet $rect avec la boucle foreach()*

Le comportement par défaut consiste à passer, à chaque itération de la boucle, d'un attribut `visible` à l'autre.

PHP vous permet d'implémenter l'interface `Iterator` afin de définir vous-même le comportement de la boucle `foreach()` :

Tableau 13.1 : *Différentes fonctions devant être implémentées pour définir votre propre itérateur.*

Méthode	Rôle
`rewind()`	Revient au début de la liste
`current()`	Valeur de l'élément en cours
`key()`	Valeur de la clé de l'élément en cours
`next()`	Passer à l'élément suivant
`valid()`	Retourne `false` lorsque vous êtes à la fin de la ligne (`true` sinon)

L'exemple suivant permet d'itérer parmi les différentes valeurs de l'attribut `$tab` plutôt que parmi les attributs de l'objet.

Listing 13-29 : Définition de votre propre itérateur

```
class Rectangle implements Iterator{

  public $longueur = null;
  public $largeur = null;
  public $couleur = "rouge";
  public $tab = array("mm","cm","m");
  public $i = 0;

  function __construct($longueur,$largeur) {
    $this->longueur = $longueur;
    $this->largeur = $largeur;
  }

  public function rewind() {
    print("rewind<br/>");
    $this->i = 0;
  }

  public function current() {
    print("current<br/>");
    return ($this->tab[$this->i]);
  }

  public function key() {
    print("key<br/>");
    return ($this->i);
  }

  public function next() {
    print("next<br/>");
    $this->i++;
  }

  public function valid() {
    print("valid<br/>");
    if ($this->i<count($this->tab)) return true;
    return false;
  }

}

$rect = new Rectangle(3,5);

foreach ($rect as $attr => $val) {
  print("<b>$attr = $val</b><br/>");
}
```

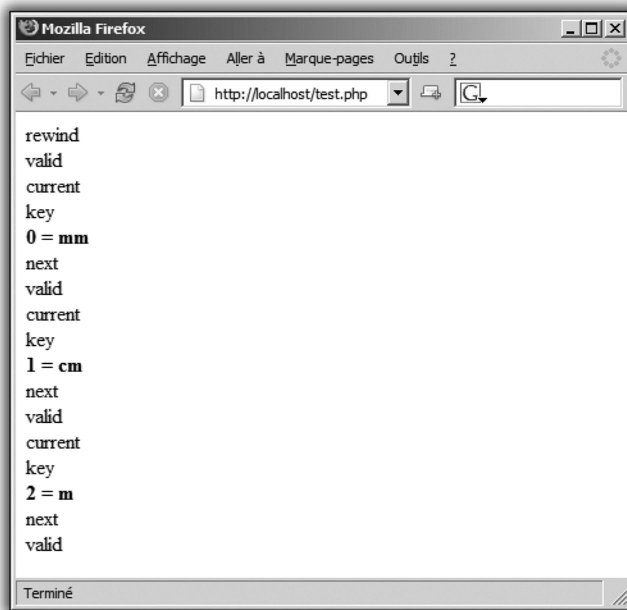


Figure 13.12 : *Trace des différents appels de PHP lors de la mise en œuvre d'un itérateur avec la boucle foreach*

13.6. Exceptions

La gestion d'erreurs en POO fait intervenir une notion supplémentaire : les exceptions. Ces dernières correspondent à des signaux pouvant être émis par une méthode et capturés ailleurs dans le code.

Principe général

L'émission d'une exception utilise le mot-clé `throw`.

Listing 13-30 : La méthode test() émet une exception lorsque $n est null

```
class MonObjet {
  function test($n=null) {
    if ($n==null) throw new Exception("La valeur ne
    ✂ convient pas");
    return true;
  }
}
```



Mot clé `throw`

L'émission d'une exception termine la méthode à la manière d'un `return ()`.

La capture d'une exception fait intervenir la structure `try { … } catch () { … }`.

Le bloc `try` permet d'englober l'ensemble des appels aux méthodes susceptibles d'émettre des exceptions. Le bloc `catch()` se charge quant à lui de recevoir l'ensemble des exceptions émises par ces méthodes. Un bloc `try` doit obligatoirement être suivi d'un bloc `catch()`.

Listing 13-31 : Contrôle d'un code susceptible d'émettre une exception

```
$obj = new MonObjet();
try {
  // utilisation d'une méthode susceptible
  // d'émettre une exception
  $obj->test();
}
catch (Exception $e) {
  // traitement de l'exception
  print($e->getMessage());
}
```



De la bonne utilisation des exceptions

Les exceptions ne doivent en aucun cas remplacer une gestion standard d'erreur. Elles ne doivent être utilisées que pour signaler des situations exceptionnelles.

L'exemple suivant montre comment émettre une exception quand la méthode `surface()` est appelée sur un rectangle invalide.

Listing 13-32 : Gestion d'une exception dans la méthode surface()

```
class Rectangle {

  public $longueur = null;
  public $largeur = null;

  function __construct($longueur,$largeur) {
    $this->longueur = $longueur;
    $this->largeur = $largeur;
  }
```

```
  function surface() {
    if ($this->longueur<1 ||
        $this->largeur<1) {
      throw new Exception("Valeurs incohérentes");
    }
    return ($this->longueur*$this->largeur);
  }

}

$rect = new Rectangle(-1,4);
try {
  echo $rect->surface();
}
catch (Exception $e) {
  print("<hr/>".$e->getMessage()."<hr/>");
}
```

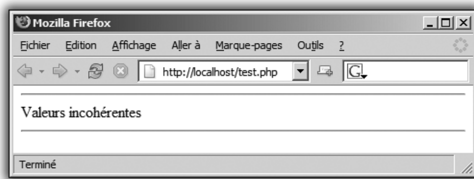


Figure 13.13 : *Affichage de l'exception*

La classe Exception

Une exception correspond à une instance de la classe `Exception`. Chaque nouvelle instance peut être initialisée avec un ou deux arguments. Le premier argument correspond au texte du message d'erreur et le second, optionnel, au code d'erreur.

Cette classe dispose d'un certain nombre de méthodes pouvant être très utiles dans une phase de débogage.

Tableau 13.2 : *Méthodes proposées par la classe Exception*

Méthode	Rôle
`getMessage()`	Retourne la valeur du message
`getCode()`	Retourne la valeur du code d'erreur
`getFile()`	Retourne le nom du fichier d'où l'exception a été émise

Tableau 13.2 : Méthodes proposées par la classe Exception	
Méthode	**Rôle**
`getLine()`	Retourne le numéro de la ligne où l'exception a été émise
`getTrace()`	Retourne un tableau de la trace d'erreur
`getTraceAsString()`	Retourne une chaîne de caractères correspondant à la trace de l'erreur

Listing 13-33 : Utilisation de toutes les méthodes proposées par la classe Exception

```
$rect = new Rectangle(-1,4);
try {
  echo $rect->surface();
}
catch (Exception $e) {
  print("<hr/>".
    "<i>Message</i> : ".$e->getMessage()."<br/>".
    "<i>Code</i> : ".$e->getCode()."<br/>".
    "<i>File</i> : ".$e->getFile()."<br/>".
    "<i>Line</i> : ".$e->getLine()."<br/>".
    "<i>TraceAsString</i> : <br/><pre>"
    ✂ .$e->getTraceAsString()."</pre>".
    "<hr/>");
}
```

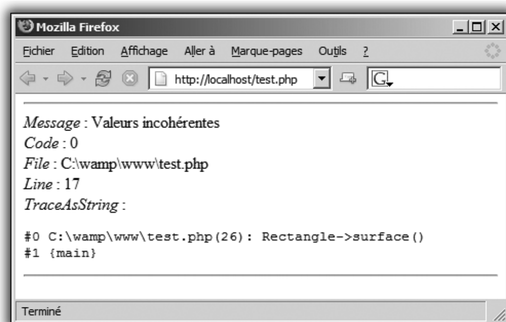


***Figure 13.14 :** Affichage des valeurs retournées par les différentes méthodes*

Cette classe peut tout à fait être étendue de façon à surcharger ses méthodes et attributs ou à les compléter avec les vôtres.

En définissant vos propres exceptions, vous vous donnez également la possibilité d'en capturer plusieurs émises dans un même bloc `try`.

```
try {
  // code susceptible d'émettre plusieurs exceptions
} catch (MonException $monexception) {
  …
} catch (Exception $exception) {
  …
}
```



Constructeur et exceptions

Un constructeur ne pouvant retourner de valeur, les exceptions sont la seule solution pour avertir d'une erreur.

13.7. Réflexion

PHP vous donne désormais la possibilité d'obtenir des informations sur la structure interne d'une classe. La classe mise en œuvre porte le nom Reflection.

```
<?php
class Rectangle {

  public $longueur = null;
  public $largeur = null;
  public $couleur = "rouge";
  const NOM = "Super Rectangle";

  function __construct($longueur,$largeur) {
    print("Hello ".self::NOM."<br/>");
    $this->longueur = $longueur;
    $this->largeur = $largeur;
  }

  function perimetre() {
    if ($this->longueur!=null &&
        $this->largeur!=null) {
      return (2*$this->longueur+2*$this->largeur);
    }
  }

  function surface() {
    if ($this->longueur!=null &&
        $this->largeur!=null) {
      return ($this->longueur*$this->largeur);
    }
  }

}
```

```
print("<pre>");
Reflection::export(new ReflectionClass('Rectangle'));
```

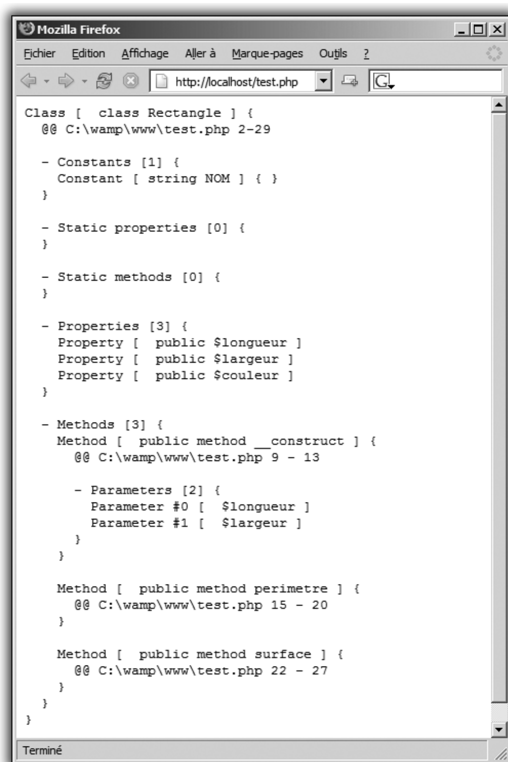


Figure 13.15 : *Affichage de la structure interne de la classe Rectangle*

Cette fonctionnalité est particulièrement intéressante pour la création automatique de documentation et pour obtenir des informations sur une classe des sources et de l'API de laquelle vous ne disposez pas.

13.8. Version objet de la génération de graphique

Le script de génération de graphique étudié dans le chapitre précédent souffre de nombreux inconvénients :

- la récupération des données n'est pas clairement séparée de l'affichage graphique ;

- les données liées à la présentation (couleurs, polices) ne sautent pas aux yeux ;
- le script n'est pas réutilisable sans un changement important du code.

Utilisez le concept de classe pour remédier à ces faiblesses. Voyez les attributs et les méthodes dont vous avez besoin.

- les attributs : toutes les couleurs, le nom de la fonte, le tableau contenant les moyennes, l'identifiant de l'image ;
- les méthodes : `__construct()`, le constructeur pour initialiser la taille de l'image et les couleurs, `enregistre_donnees()` qui permet de transmettre les données, `cree_image()` qui contient tout le code, `genere_image()` qui génère le contenu de l'image.

Il suffit ensuite de placer le code déjà écrit dans les bonnes méthodes et de passer par les attributs pour accéder aux variables :

Listing 13-34 : Le fichier class_graph.inc.php

```
<?php

class Graph {

  public $coul_fond;
  public $coul_axes;
  public $coul_lignes;
  public $coul_legendes;
  public $coul_barres;
  public $coul_orange;
  public $image;
  public $font = "verdana.ttf";
  public $abs = "trimestre";
  public $ord = "moyenne";
  public $donnees;

  public function __construct ($largeur, $hauteur,
                              $font = "arial.ttf") {
    $this->image =  Imagecreate ($largeur, $hauteur);
    if ($this->image===false) throw new Exception ();
    $this->coul_fond = ImageColorAllocate ($this->image,
                                          208, 216, 213);
    $this->coul_axes = ImageColorAllocate ($this->image,
                                          11, 62, 43);
    $this->coul_lignes = ImageColorAllocate ($this->image,
                                           227, 235, 232);
    $this->coul_legendes = ImageColorAllocate ($this->image,
                                               11, 62, 43);
```

```
    $this->coul_barres = ImageColorAllocate ($this->image,
                                              42, 124, 94);
    $this->coul_orange = ImageColorAllocate ($this->image,
                                              207, 140, 53);
    $this->font = $font;
  }

  public function enregistre_donnees ($datas) {
    $this->donnees = $datas;
  }

  public function cree_image () {
    imageline ($this->image,30,30,30,190,$this->coul_axes);
    imageline ($this->image,30,190,320,190,$this->coul _axes);

    $tab_fleche_ord = array (30, 30, 26, 34, 34, 34);
    $tab_fleche_abs = array (320, 190, 316, 186, 316, 194);

    imagefilledpolygon ($this->image, $tab_fleche_ord, 3,
                        $this->coul_axes);
    imagefilledpolygon ($this->image, $tab_fleche_abs, 3,
                        $this->coul_axes);

    ImageTTFText ($this->image,10,0,5,20,
                  $this->coul_legendes,
                  $this->font,$this->ord);
    ImageTTFText ($this->image,10,0,280,180,
                  $this->coul_legendes,
                  $this->font,$this->abs);

    imageline ($this->image,26,190,30,190,$this->coul_axes);
    imageline ($this->image,26,155,30,155,$this->coul_axes);
    imageline ($this->image,26,120,30,120,$this->coul_axes);
    imageline ($this->image,26,85,30,85,$this->coul_axes);
    imageline ($this->image,26,50,30,50,$this->coul_axes);

    ImageTTFText ($this->image,8,0,6,190,
                  $this->coul_legendes, $this->font,"0");
    ImageTTFText ($this->image,8,0,6,155,
                  $this->coul_legendes, $this->font,"5");
    ImageTTFText ($this->image,8,0,6,120,
                  $this->coul_legendes, $this->font,"10");
    ImageTTFText ($this->image,8,0,6,85,
                  $this->coul_legendes, $this->font,"15");
    ImageTTFText ($this->image,8,0,6,50,
                  $this->coul_legendes, $this->font,"20");

    imageline ($this->image,31,155,320,155,
               $this->coul_lignes);
    imageline ($this->image,31,120,320,120,
               $this->coul_lignes);
```

```
    imageline ($this->image,31,85,320,85,
               $this->coul_lignes);
    imageline ($this->image,31,50,320,50,
               $this->coul_lignes);

    imagefilledrectangle ($this->image, 40,
                          (20-$this->donnees[0])*7+50, 110,
                          189, $this->coul_barres);
    imagefilledrectangle ($this->image, 120,
                          (20-$this->donnees[1])*7+50, 190,
                          189, $this->coul_barres);
    imagefilledrectangle ($this->image, 200,
                          (20-$this->donnees[2])*7+50, 270,
                          189, $this->coul_barres);

    ImageTTFText ($this->image,10,0,50,180,$this->coul_orange,
                  $this->font,$this->donnees[0]);
    ImageTTFText ($this->image,10,0,130,180,$this->coul_orange,
                  $this->font,$this->donnees[1]);
    ImageTTFText ($this->image,10,0,210,180,$this->coul_orange,
                  $this->font,$this->donnees[2]);
  }

  public function genere_image () {
    header ("Content-type: image/png");
    ImagePng ($this->image);
  }
}

?>
```

Deux remarques peuvent être faites sur le code…

- observez la signature assez étrange du constructeur : `function __construct($largeur,$hauteur,$font = "arial.ttf")`. Cela signifie que l'on peut créer un objet `class` de deux manières :

  ```
  $mongraph = new Graph (340,220);
  ```

Dans ce cas, la variable `font` prend la valeur `"arial.ttf"`.

```
$mongraph = new Graph (340,220,"verdana.ttf");
```

Dans ce cas, l'attribut `font` est initialisé à `"verdana.ttf"`.

Pour obtenir ce résultat, vous pouvez aussi écrire :

```
$mongraph = new Graph (340,220);
$mongraph->font = "verdana.ttf";
```

- il n'est pas nécessaire que les variables `$largeur` et `$hauteur` existent en tant qu'attributs car vous n'en avez besoin que dans le constructeur.

Grâce à cette classe, le script *graph.php* se simplifie énormément :

Listing 13-35 : Le script graph.php

```
include("variables.inc.php");

include("class_graph.inc.php");

$liendb = mysql_connect($bddserver, $bddlogin,
                        $bddpassword);
mysql_select_db ($bdd);
$sql = "SELECT * FROM $table_exam";
$resultat = mysql_query ($sql);
$i = 0;
while ($tmp = mysql_fetch_array ($resultat)) {
  $tab[$tmp['trimestre'] - 1] += $tmp['moyenne'];
  $i++;
}
mysql_close($liendb);

$nb_eleves = $i / 3;

for ($i = 0; $i < 3; $i++) {
  $tab[$i] = number_format($tab[$i] / $nb_eleves,2);
}


try {
  $mongraph = new Graph(340,220);
  $mongraph->enregistre_donnees($tab);
  $mongraph->cree_image();
  $mongraph->genere_image();
}
catch (Exception $e) { die ('erreur'); }
```



Le script *class_graph.inc.php*

N'oubliez pas d'inclure le fichier *class_graph.inc.php* sous peine d'erreur !

Vous voyez que ce nouveau code présente l'avantage d'être beaucoup plus clair. Les parties acquisition des données et affichage de l'image sont désormais clairement séparées et le code est parfaitement factorisé.

Créez maintenant un fichier *graph_eleve.php* qui permettra d'afficher l'évolution des notes d'un élève :

Listing 13-36 : Le fichier graph_eleve.php

```
include("variables.inc.php");
include("class_graph.inc.php");

$liendb = mysql_connect($bddserver, $bddlogin,
                        $bddpassword);
mysql_select_db ($bdd);

$sql = "SELECT * FROM $table_exam ".
       "WHERE ideleve = '".$_REQUEST['id']."'";

$resultat = mysql_query ($sql);

while ($tmp = mysql_fetch_array ($resultat)) {
  $tab[$tmp['trimestre'] - 1] = $tmp['moyenne'];
}

mysql_close($liendb);

try {
  $mongraph = new Graph (340,220,"verdana.ttf");
  $mongraph->enregistre_donnees($tab);
  $mongraph->ord = "moyenne de l'élève";
  $mongraph->cree_image();
  $mongraph->genere_image();
}
catch (Exception $e) { die ('erreur'); }
```

Dans ce script, vous utilisez la fonte `verdana.ttf` plutôt que la fonte `arial.ttf`. De plus, vous changez l'intitulé de l'ordonnée en `"moyenne de l'élève"`. C'est possible car vous avez fait en sorte de placer de nombreuses données en attributs. Vous voyez également l'intérêt de diviser la génération d'images en plusieurs méthodes et de ne pas avoir l'intégralité du code dans le constructeur.



Envoi des fontes

N'oubliez pas d'envoyer sur votre compte les fontes que vous utilisez dans vos scripts.

Incluez la balise suivante en bas du script *eleve_edite.php* :

```
<img src="graph_eleve.php?id=<?php echo $id; ?>" />
```

Vous obtenez le résultat suivant :

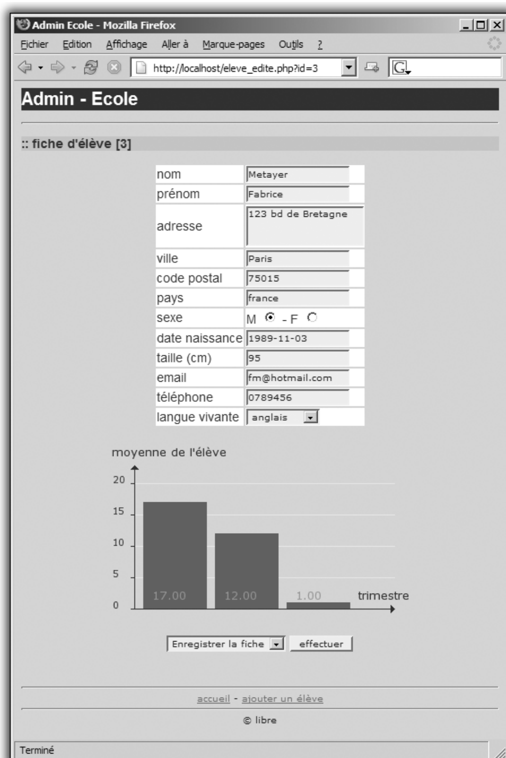


Figure 13.16 : *Une nouvelle version de la fiche élève*

13.9. Check-list

- La dimension objet de PHP a largement évolué avec la version 5.
- Une classe peut être considérée comme une entité regroupant les données et les fonctions qui lui sont propres.
- Les classes, par leur nature, sont particulièrement faciles à inclure dans une application. C'est aujourd'hui principalement sous cette forme que l'on trouve le code PHP sur le Web.
- Les techniques permettant de mieux organiser votre code sont au nombre de trois : les `include()`, les fonctions et les classes.

Chapitre 14

XML

Le format 418
SimpleXML 421
Formats spéciaux 426
Check-list 437

Peu de formats de documents ont autant influencé l'industrie informatique que le XML (*Extensible Markup Language*). Datant du milieu des années 1990, ce format est aujourd'hui la référence pour les échanges de données notamment sur Internet.

Les avantages de ce format sont nombreux :

- Un document XML est lisible, structuré et compréhensible.
- Le format respecte un standard et assure, par là même, une véritable portabilité et pérennité des données.
- Tous les langages de programmation proposent désormais des bibliothèques permettant un accès facile à ce format.

14.1. Le format

À la manière du HTML, qui comme lui est issu du SGML, le format XML s'organise avec des balises. Cependant, à la différence de HTML, les balises ne sont pas prédéfinies et peuvent être choisies en fonction des besoins de l'application. Le document suivant par exemple représente une classe :

```
<classe>
 <eleve>Paul Dupont</eleve>
 <eleve>Eric Durant</eleve>
</classe>
```

La hiérarchie saute aux yeux : une classe est composée d'élèves. Il est également possible de dire que l'élément `<classe>` dispose de plusieurs enfants : les éléments `<eleve>`.

Cette hiérarchie peut être affinée si le besoin se fait ressentir. Les nom et prénom d'un élève peuvent faire l'objet d'éléments spécifiques :

```
<classe>
 <eleve>
  <nom>Dupont</nom>
  <prenom>Paul</prenom>
 </eleve>
 <eleve>
  <nom>Durant</nom>
  <prenom>Eric</prenom>
 </eleve>
</classe>
```



Commentaires

Tout comme pour l'HTML les commentaires sont entourés des balises <!–– et ––>.

Tout comme les balises HTML, les balises XML peuvent également disposer d'attributs. Ces derniers sont, eux aussi, librement définissables. Le niveau de la classe peut être précisé de la manière suivante :

```
<classe niveau="cp">
 <eleve>
  <nom>Dupont</nom>
  <prenom>Paul</prenom>
 </eleve>
 <eleve>
  <nom>Durant</nom>
  <prenom>Eric</prenom>
 </eleve>
</classe>
```

Les documents XML précédents sont pour l'instant incorrects. La première ligne d'un document XML doit en effet préciser deux informations :

- la version de la norme XML ;
- l'encodage des données.

Cette ligne prend la forme suivante :

```
<?xml version="1.0" encoding="UTF-8"?>
```

Cette gestion de l'encodage permet à XML de contenir des données écrites dans toutes les langues. L'encodage UTF-8 notamment pourrait être assimilé à un alphabet universel des caractères de toutes les langues mondiales. Pour des langues européennes, l'encodage ISO-8859-1 peut suffire. Cet encodage permet en effet à un document XML de contenir des données accentuées.

D'autres contraintes doivent être respectées pour obtenir un document valide :

- Le document ne doit contenir qu'une racine (dans cet exemple, il s'agit de `<classe>`). Un document XML dispose d'une véritable structure d'arbre composée d'éléments (nœuds) pouvant avoir des enfants.
- Les balises orphelines doivent contenir une barre oblique (/) avant le chevron final > (par exemple `<voyages />`).

- Tous les attributs doivent être entourés de guillemets ou de primes (", ’).
- La casse doit être respectée entre une ouverture et une fermeture de balise.
- Certains caractères doivent être convertis au sein du contenu d’un élément (<, >, ’, ", & sont remplacés respectivement par <, >, ’, ’, &). La fonction PHP `htmlspecialchars()` permet de réaliser cette conversion.
- Les balises d’ouverture et de fermeture ne peuvent s’entremêler (exemple qui ne fonctionne pas : `<eleve><nom>Dupont </eleve></nom>`).

La fonction `header()` est une nouvelle fois nécessaire afin d’indiquer au navigateur que le document reçu est de type XML et qu’il doit par conséquent l’afficher dans une « vue » adaptée. Le script suivant permet d’obtenir les élèves dans le cadre d’un document XML.

Listing 14-1 : Génération d’un document XML

```
include("variables.inc.php");

$liendb = mysql_connect($bddserver, $bddlogin,
✂ $bddpassword);
mysql_select_db ($bdd);
$sql = "SELECT * FROM $table_eleve";
$resultat = mysql_query ($sql);

header(’Content-Type: application/xml’);
$xml = ’<?xml version="1.0" encoding="ISO-8859-1"?>’;
$xml .= "\n<classe>\n";
while ($eleve = mysql_fetch_array ($resultat)) {
  foreach ($eleve as &$valeur) {
    $value = htmlspecialchars($value);
  }
  $xml .= " <eleve id=’".$eleve[’ideleve’]."’>\n";
  $xml .= "  <nom>".$eleve[’nom’]."</nom>\n";
  $xml .= "  <prenom>".$eleve[’prenom’]."</prenom>\n";
  $xml .= "  <adresse>".$eleve[’adresse’]."</adresse>\n";
  $xml .= "  <ville>".$eleve[’ville’]."</ville>\n";
  $xml .= "  <cp>".$eleve[’cp’]."</cp>\n";
  $xml .= "  <pays>".$eleve[’pays’]."</pays>\n";
  $xml .= " </eleve>\n";
}
mysql_close($liendb);
$xml .= "</classe>\n";

print($xml);
```

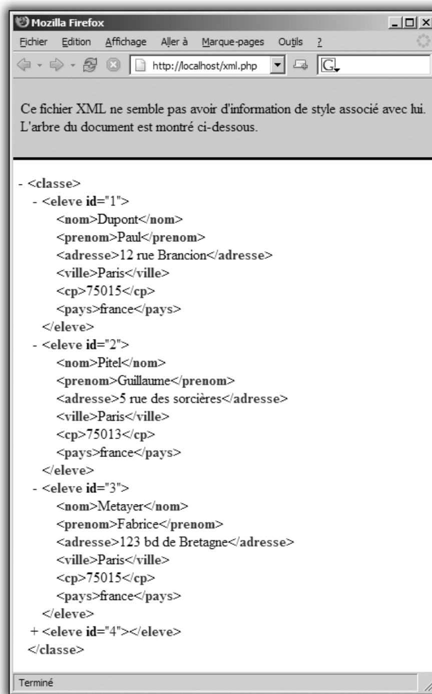


Figure 14.1 : *Affichage des données en mode XML dans Firefox*



CDATA

Il est possible de définir un contenu d'élément sans se préoccuper des contraintes décrites plus haut. La section d'échappement spéciale suivante `<![CDATA[ ]]>` permet d'indiquer que tout ce qui est placé entre `<![CDATA[` et `]]>` ne doit pas être interprété mais considéré comme du texte brut.

L'écriture suivante devient donc possible

```
<test><![CDATA[pas de problème de balise orpheline :
<img>]]></test>
```

14.2. SimpleXML

Au lieu de manipuler directement les balises comme dans les exemples précédents, PHP propose l'extension SimpleXML pour construire un

document XML mais aussi et surtout le lire et accéder directement à n'importe quel nœud du document.

Autre avantage non négligeable de SimpleXML, cette extension est incluse par défaut dans PHP5.

Création

La création d'un document XML avec SimpleXML est très simple. Le point de départ consiste à créer l'élément racine ; un élément est représenté par la classe `SimpleXMLElement`. Le constructeur de cette classe prend en argument une chaîne de caractères représentant une donnée XML valide.

```
$racine = new SimpleXMLElement('<classe/>');
```

L'étape suivante consiste à attacher des éléments enfants à cette racine à l'aide de la méthode `addChild()` qui prend en premier argument le nom de l'élément et en second argument la valeur de l'élément.

```
$racine->addChild('eleve', 'Paul Dupont');
$racine->addChild('eleve', 'Eric Durant');
```

La représentation textuelle peut ensuite être obtenue en faisant appel à la méthode `asXML()` de l'élément racine (`$classe`).

```
echo $racine->asXML();
```

Cette méthode retourne une chaîne de caractères et ne prend en aucun cas soin de modifier le `Content-Type` pour indiquer au navigateur que le contenu est de type XML. La fonction `header()` doit par conséquent être utilisée.

Listing 14-2 : Création d'un document XML à l'aide de l'extension SimpleXML

```
$racine = new SimpleXMLElement('<classe/>');
$racine->addChild('eleve', 'Paul Dupont');
$racine->addChild('eleve', 'Eric Durant');
header('Content-Type: application/xml');
echo $racine->asXML();
```

SimpleXML permet également d'ajouter des attributs aux éléments à l'aide de la méthode `addAttribute()`.



Figure 14.2 : *La méthode asXML() retourne bien une chaîne de caractères correspondant à la représentation XML*

Listing 14-3 : Utilisation de la méthode addAttribute()

```
$eleves = array(array('id' => 1,
                      'nom' => 'gueudet',
                      'prenom' => 'edouard'),
                array('id' => 2,
                      'nom' => 'henry',
                      'prenom' => 'thomas'));
$racine = new SimpleXMLElement('<classe/>');
foreach ($eleves as $eleve) {
  $noeud = $racine->addChild('eleve');
  $noeud->addChild('nom', $eleve['nom']);
  $noeud->addChild('prenom', $eleve['prenom']);
  $noeud->addAttribute('id', $eleve['id']);
}
header('Content-Type: application/xml');
echo $racine->asXML();
```

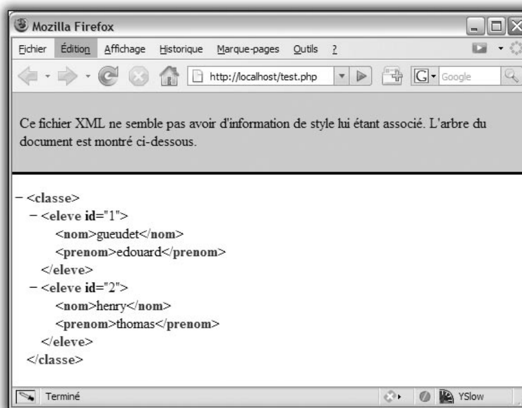


Figure 14.3 : *L'attribut id a bien été ajouté aux éléments <eleve>*

SimpleXML (comme les autres extensions PHP relatives à XML) manipule en interne des chaînes de caractères au format UTF-8. Il convient donc de vérifier que les données transmises sont au bon format.

Listing 14-4 : Utilisation de caractères non ASCII

```
$racine = new SimpleXMLElement('<tests/>');
header('Content-Type: application/xml');
$racine->addChild('test', 'hélène');
echo $racine->asXML();
```

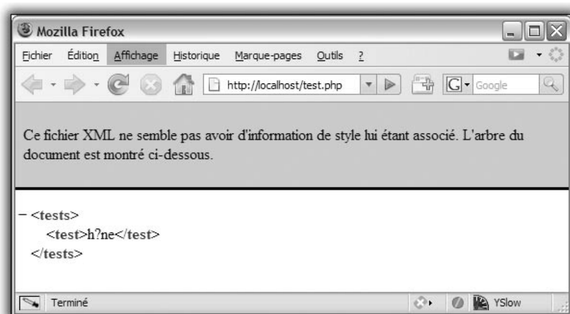


Figure 14.4 : La chaîne "hélène", non convertie, s'affiche mal

Deux solutions permettent de résoudre cette problématique. La première consiste à utiliser la fonction `utf8_encode()` pour transmettre toutes les chaînes dans le bon encodage.

```
$racine->addChild('test', utf8_encode('hélène'));
```

La seconde repose sur l'éditeur utilisé pour écrire les sources du script. Si le format d'encodage sélectionné est bien UTF-8, la donnée apparaîtra alors sans erreur.

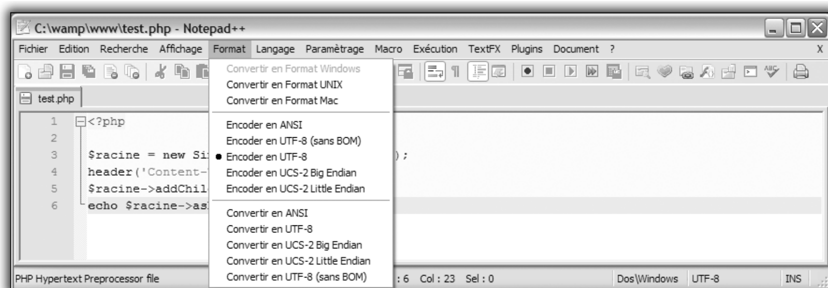


Figure 14.5 : Notepad++ permet de forcer l'encodage du script en UTF8



Figure 14.6 : *Le prénom "hélène" apparaît sans erreur quand l'encodage UTF8 est utilisé*

Lecture

L'opération de lecture repose en grande partie sur la méthode `children()`. Cette dernière retourne l'ensemble des enfants d'un élément donné. Fonctionnalité extrêmement pratique, un élément (`SimpleXMLElement`) donne accès aux valeurs de ses enfants directement via ses attributs. L'exemple suivant montre comment un élément (`coords`) peut accéder à la valeur de ses deux enfants : `x` et `y`.

```
$coords = new SimpleXMLElement('<coords><x>2</x><y>3</y>
✂ </coords>');
echo "x=".$coords->x." y=".$coords->y;
```

L'affichage des noms et prénoms des élèves d'une classe devient donc un jeu d'enfant.

Listing 14-5 : Parcours des élèves de la classe

```
$str = '<classe>
  <eleve id="1">
    <nom>gueudet</nom>
    <prenom>edouard</prenom>
  </eleve>
  <eleve id="2">
    <nom>henry</nom>
    <prenom>thomas</prenom>
  </eleve>
</classe>';

$classe = new SimpleXMLElement($str);

$eleves = $classe->children();
```

```
foreach ($eleves as $eleve) {
  echo $eleve->prenom." ".$eleve->nom."<br/>";
}
```

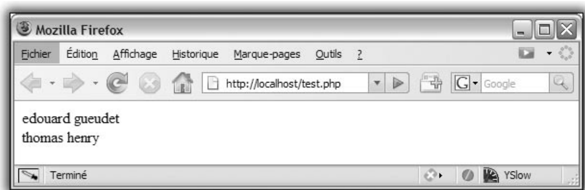


Figure 14.7 : *Les 2 élèves ont bien été trouvés*

La méthode `attributes()` donne accès à l'ensemble des attributs d'un élément.

L'exemple précédent peut être modifié de la façon suivante pour afficher l'`id` de l'élève.

```
foreach ($eleves as $eleve) {
  $attributs = $eleve->attributes();
  echo "[".$attributs["id"]."] ";
  echo $eleve->prenom." ".$eleve->nom."<br/>";
}
```

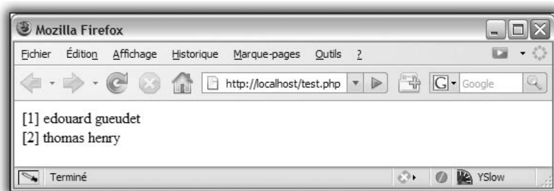


Figure 14.8 : *Affichage de l'id*

14.3. Formats spéciaux

Le format XML permet d'organiser les données selon son bon vouloir tout en étant certain qu'un partenaire sera en mesure de le manipuler. Il conviendra cependant, en même temps que la transmission des données, de fournir une documentation décrivant comment sont agencés les attributs et les balises, à quoi ils correspondent, etc.

Pour éviter cette fastidieuse tâche de description, l'industrie s'est très vite organisée pour faire émerger des structures de documents XML prédéfinies (spécifications).

RSS

Le format RSS est un cas typique de standardisation d'une structure XML pour répondre à un besoin partagé par de nombreux intervenants du monde du web. L'idée est ici de permettre à un site de contenu de pouvoir fournir à des partenaires un document contenant un résumé des derniers articles mis en ligne. Ces documents sont généralement qualifiés de flux (*feed*) RSS.



Figure 14.9 : *Icône utilisée par les différents navigateurs pour indiquer la présence d'un flux RSS*

En établissant ce standard, l'industrie informatique a pu miser et investir dessus. Les navigateurs web peuvent directement les afficher, des appareils électroniques tels que des iPods sont en mesure d'importer leur contenu, des sites web tels que Google Reader ne servent qu'à les agréger pour pouvoir les consulter en un point central.

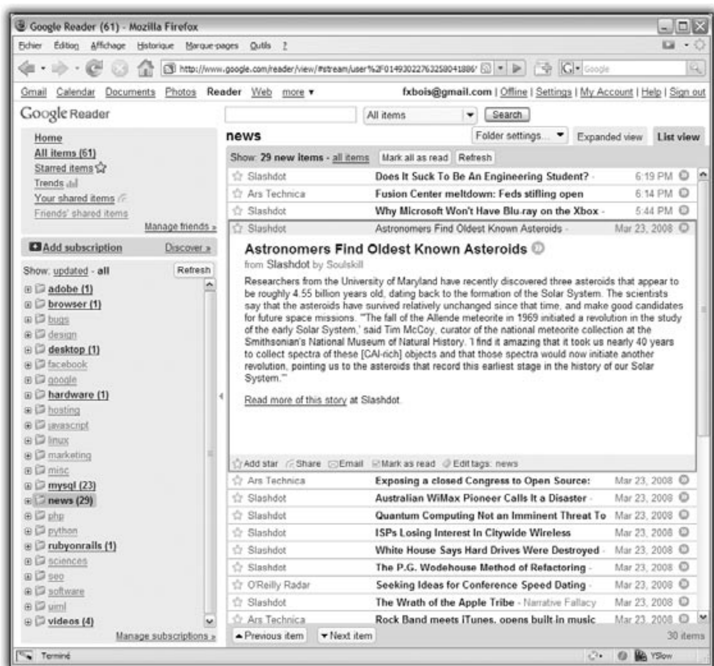
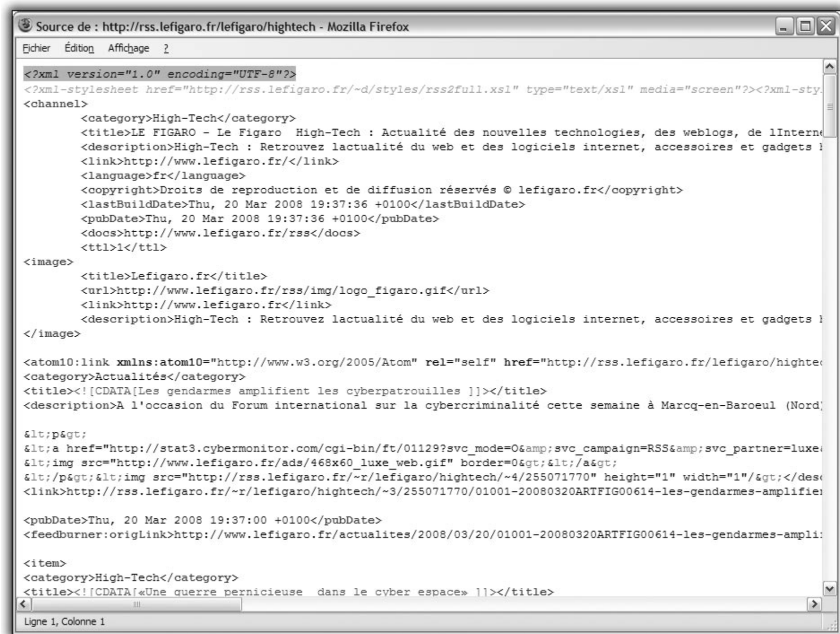


Figure 14.10 : *Google Reader, lecteur de flux RSS*



Figure 14.11 : *Affichage du flux RSS High Tech du Figaro*




```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="http://rss.lefigaro.fr/~d/styles/rss2full.xsl" type="text/xsl" media="screen"?><?xml-sty
<channel>
        <category>High-Tech</category>
        <title>LE FIGARO - Le Figaro  High-Tech : Actualité des nouvelles technologies, des weblogs, de lInterne
        <description>High-Tech : Retrouvez lactualité du web et des logiciels internet, accessoires et gadgets
        <link>http://www.lefigaro.fr/</link>
        <language>fr</language>
        <copyright>Droits de reproduction et de diffusion réservés © lefigaro.fr</copyright>
        <lastBuildDate>Thu, 20 Mar 2008 19:37:36 +0100</lastBuildDate>
        <pubDate>Thu, 20 Mar 2008 19:37:36 +0100</pubDate>
        <docs>http://www.lefigaro.fr/rss</docs>
        <ttl>1</ttl>
<image>
        <title>Lefigaro.fr</title>
        <url>http://www.lefigaro.fr/rss/img/logo_figaro.gif</url>
        <link>http://www.lefigaro.fr</link>
        <description>High-Tech : Retrouvez lactualité du web et des logiciels internet, accessoires et gadgets
</image>

<atom10:link xmlns:atom10="http://www.w3.org/2005/Atom" rel="self" href="http://rss.lefigaro.fr/lefigaro/hightec
<category>Actualités</category>
<title><![CDATA[Les gendarmes amplifient les cyberpatrouilles ]]></title>
<description>A l'occasion du Forum international sur la cybercriminalité cette semaine à Marcq-en-Baroeul (Nord)

&lt;p&gt;
&lt;a href="http://stat3.cybermonitor.com/cgi-bin/ft/01129?svc_mode=O&amp;svc_campaign=RSS&amp;svc_partner=luxe
&lt;img src="http://www.lefigaro.fr/ads/468x60_luxe_web.gif" border=0&gt;&lt;/a&gt;
&lt;/p&gt;&lt;img src="http://rss.lefigaro.fr/~r/lefigaro/hightech/~4/255071770" height="1" width="1"/&gt;</desc
<link>http://rss.lefigaro.fr/~r/lefigaro/hightech/~3/255071770/01001-20080320ARTFIG00614-les-gendarmes-amplifie

<pubDate>Thu, 20 Mar 2008 19:37:00 +0100</pubDate>
<feedburner:origLink>http://www.lefigaro.fr/actualites/2008/03/20/01001-20080320ARTFIG00614-les-gendarmes-ampli

<item>
<category>High-Tech</category>
<title><![CDATA[«Une guerre pernicieuse  dans le cyber espace» ]]></title>
```

Figure 14.12 : *L'affichage des sources prouve bien qu'il s'agit d'un document XML de type RSS*

Structure

L'élément de plus haut niveau est la balise `<rss>` incluant en attribut la version du format. La version la plus récente et la plus répandue est la 2.0 : `<rss version= "2.0">`.

Le seul enfant de l'élément `<rss>` est l'élément `<channel>`.

Tableau 14.1 : *Enfants de l'élément* `channel`		
Elément	**Usage**	**Remarque**
`title`	Titre du flux	Obligatoire
`link`	Url permettant d'accéder à ce flux	Obligatoire
`description`	Description	Obligatoire
`language`	Langue	

Tableau 14.1 : Enfants de l'élément channel		
Elément	**Usage**	**Remarque**
Image	Image	
copyright	Copyright	
managingEditor	Adresse email de la personne responsable du flux	
webMaster	Adresse email du webmaster	
pubDate	Date de publication	Format RFC 822
lastBuildDate	Date de dernière génération	Format RFC 822
rating		
category	Catégories	
generator	Nom du logiciel qui a permis de construire le flux	
ttl	Durée de vie	en minutes

L'élément <channel> peut disposer également de plusieurs éléments enfants <item> qui correspondent véritablement aux histoires/articles associés au flux.

Un <channel> contient donc plusieurs <item>. Une analogie possible consisterait à comparer un <channel> à un journal et les <item> à des articles. Le journal dispose bien d'un titre et d'un rédacteur en chef, comme les articles ont un titre et un auteur.

Chaque <item> dispose de plusieurs éléments enfants.

Tableau 14.2 : Enfants de l'élément item		
Elément	**Usage**	**Remarque**
title	Titre	Obligatoire
link	Lien direct vers l'article	Obligatoire
description	Description	Obligatoire
author	Auteur	Adresse email

Tableau 14.2 : Enfants de l'élément item		
Elément	**Usage**	**Remarque**
category	Catégories	
comments	Lien vers la page de commentaires	
guid	Numéro/Code unique de l'article	
pubDate	Date de publication	RFC 822

Le script suivant liste les enfants d'une classe mais cette fois sous la forme d'un flux RSS.

Listing 14-6 : Génération d'un flux RSS

```
$classe = array(array('id' => 1,
                      'nom' => 'gueudet',
                      'prenom' => 'edouard'),
                array('id' => 2,
                      'nom' => 'henry',
                      'prenom' => 'thomas'));

$rss = new SimpleXMLElement('<rss/>');
$rss->addAttribute('version', '2.0');

$channel = $rss->addChild('channel');
$channel->addChild('title', 'la classe');
$channel->addChild('link', 'http://localhost/test.php');
$channel->addChild('description', 'élèves de la classe');
$channel->addChild('generator', 'a la mano');

$lien_edit = 'http://localhost/eleve_edite.php?id=';
foreach ($classe as $eleve) {
  $item = $channel->addChild('item');
  $item->addChild('title', $eleve['prenom'].'
  ✂ '.$eleve['nom']);
  $item->addChild('link', $lien_edit.$eleve['id']);
  $item->addChild('guid', 'eleve-'.$eleve['id']);
  $item->addChild('description', 'id='.$eleve['id']);
}

header('Content-Type: application/xml');
echo $rss->asXML();
```

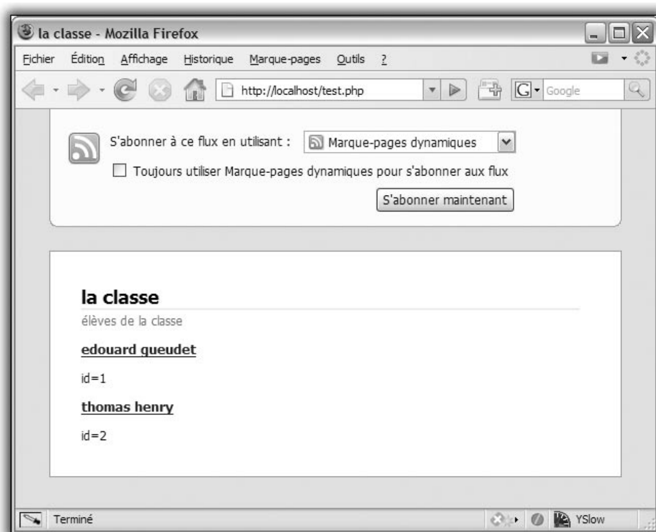


Figure 14.13 : *L'affichage d'un document XML est spécial dans le cadre d'un flux RSS*

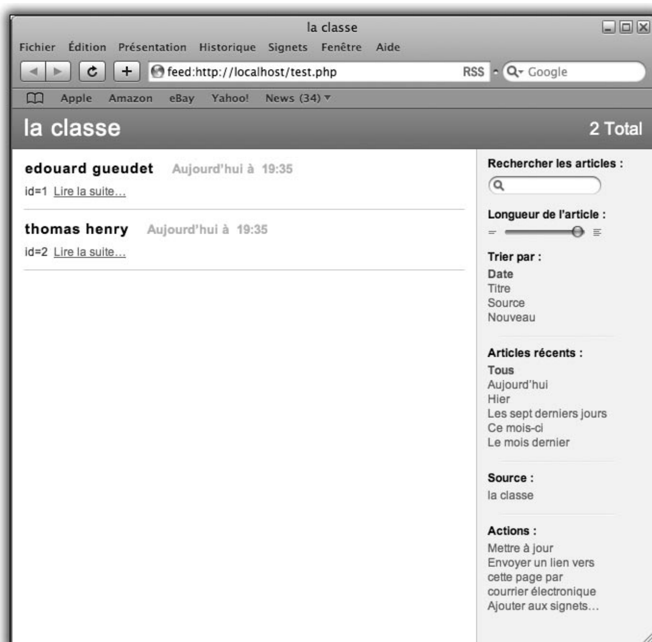


Figure 14.14 : *Chaque navigateur a un mode d'affichage des flux XML qui lui est propre*

Inclusion dans un site

Tous les navigateurs sont aujourd'hui en mesure d'indiquer la présence d'un flux RSS sur un site web. Pour Firefox, il s'agit d'une petite icône orange située à droite de l'adresse du site.



Figure 14.15 : *Le site kernix.com dispose bien d'un flux RSS associé*

Le site indique la présence d'un flux en utilisant une balise `<LINK>` spécifique dans l'en-tête (`<HEADER>`) de la page.

Listing 14-7 : Balise à ajouter pour indiquer au navigateur que le site dispose d'un flux RSS

```
<link rel="alternate" type="application/rss+xml"
✂ title="RSS" href="/test.php" />
```

L'attribut `href` permet de préciser l'emplacement du flux.



Flux multiples

Il est possible d'associer plusieurs flux à une page en multipliant les balises `<LINK>` dans l'en-tête. En cliquant sur l'icône de flux, le navigateur propose alors de sélectionner le flux souhaité. L'attribut `title` est dans ce cas très utile.

XHTML

L’XHTML est une version de l’HTML pour laquelle les contraintes de l’XML sont respectées. En développant des pages à la norme XHTML, ces dernières deviennent directement manipulables comme tout autre document XML.

L’exemple suivant permet de récupérer tous les liens (`<a href="">`) de la page du W3C consacrée à l’XHTML : **www.w3.org/TR/xhtml1**.

Le constructeur de la classe `SimpleXMLElement` peut prendre en premier paramètre une URL correspondant à un document XML. Pour cela, le troisième paramètre vaut `true`, et le second, `null`.

```
$liens = array();

function extraitLiens($noeud) {
  if (strtoupper($noeud->getName()) == 'A') {
    $attributs = $noeud->attributes();
    $href = (string) $attributs['href'];
    if (strlen($href) > 1) {
      $GLOBALS['liens'][] = $href;
    }
  }
  foreach ($noeud->children() as $enfant) {
    extraitLiens($enfant);
  }
}

$xml = new SimpleXMLElement("http://www.w3.org/TR/xhtml1/",
                            null,
                            true);

extraitLiens($xml);

print("<pre>");
print_r($liens);
print("</pre>");
```

Le script repose sur une utilisation récursive de la fonction `extraitLiens()`, qui, pour chaque nœud reçu en paramètre, teste si ce dernier correspond à un lien (`== 'A'`) et s’appelle de nouveau pour tous les éventuels enfants.

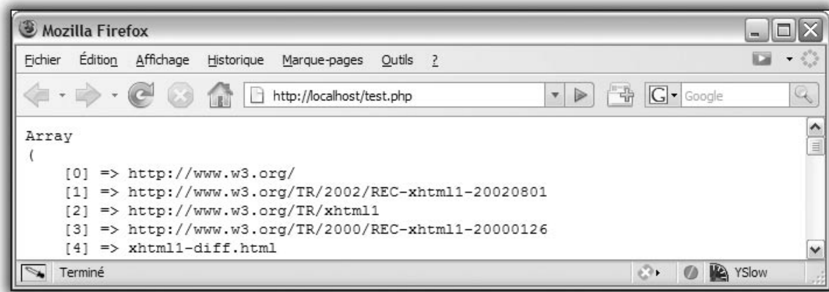


Figure 14.16 : *Liste des liens*

SVG

SVG est une spécification XML permettant de construire des images au format vectoriel. Une telle image a l'avantage de pouvoir être déformable sans perte de qualité.

Parmi les logiciels graphiques, Illustrator appartient au monde du vectoriel et Photoshop à celui des images bitmap.

L'exemple suivant permet de construire pas à pas un échiquier de 64 pièces.

Listing 14-8 : Construction d'un échiquier

```
$nb_pieces = 8;
$taille_piece = 30;
$couleurs = array('#FFFFFF', '#000000');

$svg = new SimpleXMLElement('<svg/>');
$svg->addAttribute('width', $nb_pieces * $taille_piece);
$svg->addAttribute('heigth', $nb_pieces * $taille_piece);
$svg->addAttribute('xmlns', 'http://www.w3.org/2000/svg');

for ($i = 0; $i < $nb_pieces; $i++) {
  for ($j = 0; $j < $nb_pieces; $j++) {
    $carre = $svg->addChild('rect');
    $carre->addAttribute('x', $j*$taille_piece);
    $carre->addAttribute('y', $i*$taille_piece);
    $carre->addAttribute('width', $taille_piece);
    $carre->addAttribute('height', $taille_piece);
    $carre->addAttribute('style',
    ✂ 'fill:'.$couleurs[($i+$j)%2]);
  }
```

```
}

header('Content-Type: application/xml');
echo $svg->asXML();
```

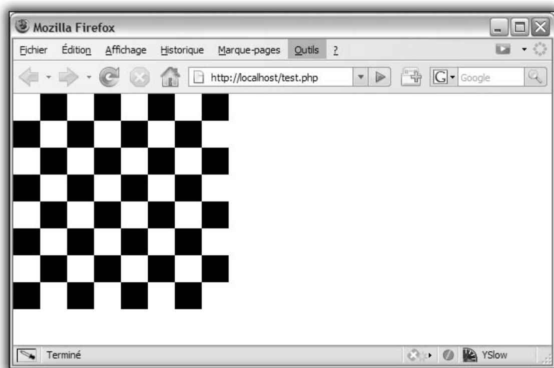


Figure 14.17 : *Affichage de l'échiquier dans un navigateur*



Ce fichier XML ne semble pas avoir d'information de style lui étant associé. L'arbre du document est montré ci-dessous.

```
– <svg width="240" heigth="240">
    <rect x="0" y="0" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="30" y="0" width="30" height="30" style="fill:#000000"/>
    <rect x="60" y="0" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="90" y="0" width="30" height="30" style="fill:#000000"/>
    <rect x="120" y="0" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="150" y="0" width="30" height="30" style="fill:#000000"/>
    <rect x="180" y="0" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="210" y="0" width="30" height="30" style="fill:#000000"/>
    <rect x="0" y="30" width="30" height="30" style="fill:#000000"/>
    <rect x="30" y="30" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="60" y="30" width="30" height="30" style="fill:#000000"/>
    <rect x="90" y="30" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="120" y="30" width="30" height="30" style="fill:#000000"/>
    <rect x="150" y="30" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="180" y="30" width="30" height="30" style="fill:#000000"/>
    <rect x="210" y="30" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="0" y="60" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="30" y="60" width="30" height="30" style="fill:#000000"/>
    <rect x="60" y="60" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="90" y="60" width="30" height="30" style="fill:#000000"/>
    <rect x="120" y="60" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="150" y="60" width="30" height="30" style="fill:#000000"/>
    <rect x="180" y="60" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="210" y="60" width="30" height="30" style="fill:#000000"/>
```

Figure 14.18 : *Les sources de la page révèlent bien qu'un document XML se cache derrière l'image*



Création d'images au format SVG

Le logiciel Inkscape est un concurrent open source d'Illustrator. Il peut être téléchargé gratuitement sur le site www.inkscape.org. Les créations réalisées peuvent être sauvegardées au format *SVG* et analysées avec un simple éditeur de texte pour comprendre la structure.

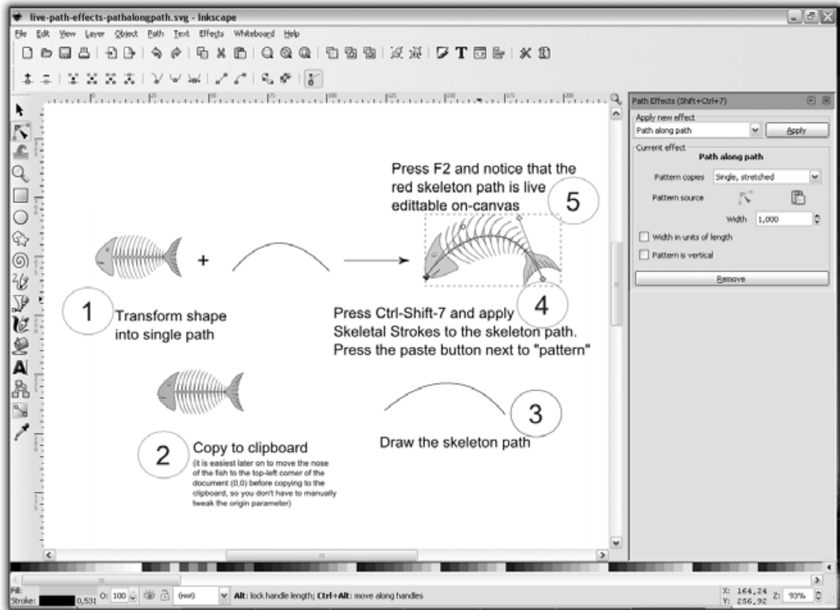


Figure 14.19 : *Inkscape permet de créer des images au format SVG*

14.4. Check-list

- Le format XML permet de stocker des informations et se révèle être le meilleur choix pour les échanges de données structurées.
- L'utilisation d'une extension de type SimpleXML est préférable à une construction "à la main" d'un fichier XML.
- Les nouveaux formats informatiques prennent de plus en plus souvent la forme de spécifications XML (*RSS*, *SVG*, *ODF*, *GDATA*, *SOAP*).

Chapitre 15

Les cookies et les sessions

Les cookies 440
Les sessions 472
Check-list 482

Nous nous intéresserons dans ce chapitre aux différentes méthodes qui sont offertes pour associer des données à un internaute : les cookies et les sessions.

Pour illustrer ces différentes techniques, vous mettrez en place une mini-boutique.

15.1. Les cookies

En surfant sur le Web, vous avez pu vous apercevoir que certains sites étaient en mesure de vous reconnaître. En revenant sur une boutique où vous avez déjà acheté, il n'est pas rare d'y trouver des messages du genre : « Bienvenue M. Dupont » ; « Vous avez acheté le produit X, nous vous proposons cette liste de produits qui peuvent vous intéresser » ; « Voici les derniers produits apparus dans notre boutique depuis votre dernière visite », etc.

En vous connectant depuis une autre machine ou un autre compte, vous vous apercevrez cependant que tous ces messages sont absents.

Ces sites utilisent une technique qui leur permet de stocker des informations dans votre ordinateur, informations qui leur sont retransmises dès que vous naviguez sur leur site. Les données sont stockées dans de petits fichiers appelés *cookies*. Ces cookies ne servent pas que les intérêts marketing des grandes boutiques du Web, ils vous permettent aussi :

- de ne pas avoir à retaper systématiquement votre identifiant et votre mot de passe sur certains sites ;
- de pouvoir disposer d'un système de panier très pratique dans les boutiques en ligne.

Ces cookies contiennent tout type d'information et sont utilisés dans de très nombreux cas de figure. Souvent diabolisés, ils vous rendent finalement davantage service qu'ils ne vous desservent. Leur présence est d'ailleurs d'autant moins gênante qu'ils peuvent être effacés à tout moment. Sous Firefox, il suffit d'aller dans le menu **Outils/Vie privée** > **Afficher les cookies** puis d'appuyer sur le bouton **Supprimer tous les cookies** pour les faire disparaître.

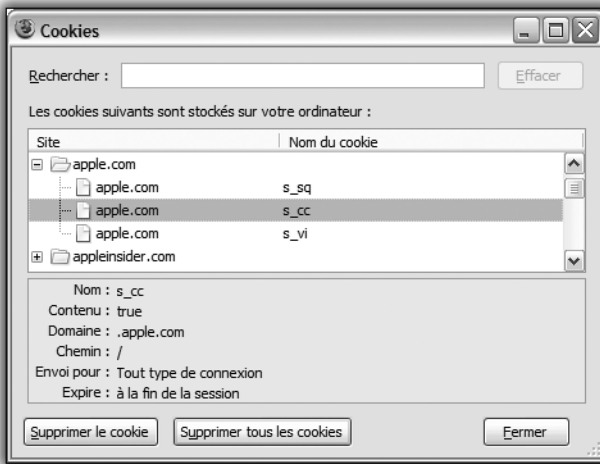


Figure 15.1 : *Suppression des cookies*

La perte de l'espace disque est aussi un reproche, très souvent exagéré, que l'on fait aux cookies. Un cookie fait en moyenne, entre 50 et 150 octets. Il vous faudrait stocker 10 millions de cookies pour perdre 1 Go d'espace disque !



Ce n'est pas gênant, mais quand même…

En identifiant de manière unique les internautes, certains services tels que doubleclik.com (gestion de bannières publicitaires) ont constitué une base de données mondiale afin d'étudier leur comportement. Ils savent où vous êtes allé, quand et pendant combien de temps… *Big Brother* n'est alors plus très loin !

Aspects techniques

Plusieurs fois dans ce livre, nous nous sommes intéressés à l'en-tête HTTP des pages web, notamment lors de l'utilisation de la fonction `header()`. Les cookies sont en réalité des données stockées en texte, qui sont transmises dans l'en-tête HTTP des requêtes et des réponses :

Listing 15-1 : Exemple de directive HTTP permettant de créer le cookie moncookie

```
Set-Cookie: moncookie=hello; path=/; expires Mon,
✂ 09-Dec-2002 13:46:00 GMT
```

Listing 15-2 : Directive HTTP contenue dans la requête envoyée par un navigateur

```
Cookie: moncookie=hello
```

La gestion des cookies en PHP est très simple. Elle ne nécessite que l'utilisation de la fonction `setcookie()`. Cette fonction prend par défaut deux paramètres : le nom du cookie et sa valeur.

Écrivez deux scripts, l'un pour créer le cookie et l'autre pour lire sa valeur :

Listing 15-3 : Le script cree_cookie.php

```
<?php
setcookie("moncookie","hello");
echo "cookie créé";
?>
```

Listing 15-4 : Le script lit_cookie.php

```
<?php
echo "valeur contenue dans mon cookie :
✂ ".$_COOKIE['moncookie'];
?>
```

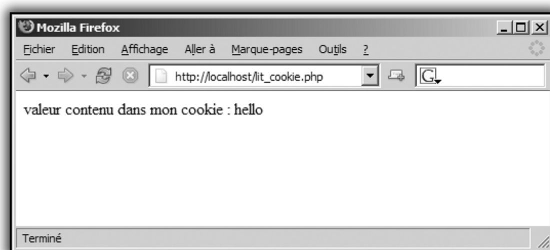


Figure 15.2 : *Lecture d'un cookie*

Comme les paramètres avec `$_REQUEST[]`, les cookies peuvent être récupérés au sein d'une variable super-globale : `$_COOKIE`. Ce tableau associatif contient autant de cellules que vous avez créé de cookies. Une fois le cookie `moncookie` créé, la variable `$_COOKIE['moncookie']` devient accessible dans vos scripts.



Suppression

Pour supprimer un cookie en PHP, il suffit de supprimer le contenu du cookie. Si vous souhaitez supprimer le cookie `moncookie`, écrivez `setcookie("moncookie")`.

Comme les cookies sont transmis dans l'en-tête HTTP, leur utilisation impose les mêmes contraintes que pour la fonction `header()` : interdiction absolue d'afficher quoi que ce soit avant d'utiliser `setcookie()`.



Limitations

Tout n'est tout de même pas permis avec les cookies. Voici une liste non exhaustive des limitations :

- Seul le site qui a créé le cookie peut y accéder.
- La taille d'un cookie doit être inférieure à 4 ko.
- Le nombre de cookies créés par un domaine donné est limité à 20.

La fonction `setcookie()` peut prendre d'autres paramètres…

- La date d'expiration : en secondes, la fonction PHP `time()` renvoyant la date actuelle en secondes.
- Le chemin : indique quelle partie du site peut avoir accès au cookie.
- Le domaine : indique quel domaine peut avoir accès au site.
- Un indicateur de sécurité : si sa valeur est `1`, il indique que la valeur du cookie ne peut être transmise que si vous êtes en HTTS (HTTP sécurisé par du SSL).

Voyez ces quelques exemples…

Premier exemple : modifiez le fichier *cree_cookie.php*.

```
<?php

setcookie("moncookie","hello",time()+5);

echo "cookie créé";

?>
```

Si vous appuyez, au bout de 5 secondes, sur le bouton **Rafraîchir** (*reload*) de la page **http://localhost/lit_cookie.php**, le cookie disparaîtra.



Heure du serveur

Il est courant que l'heure du serveur web soit décalée par rapport à l'heure de votre machine. Cela peut fausser vos tests. Pour voir la date et l'heure du serveur web, utilisez la fonction `date ("Y-m-d G:i:s")`.

Deuxième exemple :

```
setcookie("moncookie","hello",time()+3600,"/",".kernix .com",1);
```

Le cookie est, cette fois, créé pour une heure. Tous les scripts situés sur le domaine `.kernix.com` peuvent y accéder, à la condition que les transmissions soient cryptées.

Troisième exemple :

```
setcookie("moncookie","hello");
```

En ne précisant pas de date d'expiration, vous créez cette fois un cookie de session. Le cookie existera tant que le navigateur restera ouvert. Dès sa fermeture, le cookie sera automatiquement effacé.

Pour tester ce comportement, enchaînez les étapes suivantes :

1. Appelez le script *http://localhost/cree_cookie.php* (en ayant modifié son contenu au préalable !).
2. Appelez le script *http://localhost/lit_cookie.php*.
3. Fermez le navigateur.
4. Appelez de nouveau le script *http://localhost/lit_cookie.php*. Le cookie a disparu.

Pour changer la valeur du cookie `moncookie`, il suffit d'appeler la fonction `setcookie()` en modifiant la valeur du cookie.

Application : la mini-boutique FoxShop

Pour appliquer les concepts que vous venez de voir, vous allez développer une boutique extrêmement simplifiée. Les cookies seront utilisés à plusieurs endroits :

- pour la gestion du panier ;
- pour enregistrer le profil du client et lui épargner une nouvelle saisie à chaque achat.

Pour ne pas mélanger les scripts de cette boutique avec votre applicatif de gestion d'école, vous allez créer le répertoire *boutique* sur votre compte distant et y placer tous vos développements. Pour accéder à la page d'accueil de la boutique, l'URL sera donc **http://localhost/boutique**.

Au niveau de la base de données, vous avez besoin de deux tables.

- *produit :* idproduit, référence produit, nom, prix, description.
- *commande :* idcommande, liste des produits, montant global, nom client, prénom client, adresse client, date.

Votre applicatif sera composé de deux parties : front-office et back-office.

Le front-office, qui permet à un internaute d'acheter en ligne, contient les fichiers suivants :

- *index.php* (page d'accueil de la boutique) ;
- *boutique.php* (affichage du catalogue) ;
- *ajout_caddie.php* (ajout d'un produit au panier) ;
- *voir_caddie.php* (résumé de la commande, identification de paiement) ;
- *enregistre_commande.php* (enregistrement de la commande).

Le back-office, qui permet de gérer les produits et les commandes, contient les fichiers suivants :

- *adm_produits.php* (script permettant l'ajout de produits) ;
- *adm_commandes.php* (script listant les commandes) ;
- *identification.inc.php* ;
- *variables.inc.php* ;
- *haut.inc.php* ;
- *bas.inc.php*.



Entraînez-vous

Ce pourrait être le bon moment de fermer le livre et d'essayer de réaliser ce petit applicatif. Si vous cherchez de votre côté, les progrès viendront en effet bien plus vite.

Le back-office

Les tables dont vous avez besoin peuvent être définies de la manière suivante :

Table *produit* :

Listing 15-5 : Table produit

```
CREATE TABLE produit (
  idproduit int(10) unsigned NOT NULL auto_increment,
  reference varchar(16) NOT NULL default '',
  nom varchar(16) NOT NULL default '',
  description tinytext NOT NULL,
  prix float(12,2) unsigned NOT NULL default '0.00',
  PRIMARY KEY  (idproduit)
)
```

Table *commande* :

Listing 15-6 : Table commande

```
CREATE TABLE commande (
  idcommande int(10) unsigned NOT NULL auto_increment,
  produits varchar(64) NOT NULL default '',
  montant float(12,2) unsigned NOT NULL default '0.00',
  nom varchar(16) NOT NULL default '',
  prenom varchar(16) NOT NULL default '',
  adresse tinytext NOT NULL,
  date datetime NOT NULL default '0000-00-00 00:00:00',
  PRIMARY KEY  (idcommande)
)
```

Listing 15-7 : Le script adm_produits.php

```
<?php

include("variables.inc.php");
include("identification.inc.php");

if ($_REQUEST['enregistre'] == "oui") {

  if (empty($_REQUEST['reference']) ||
      empty($_REQUEST['nom']) ||
      empty($_REQUEST['prix']) ||
      empty($_REQUEST['description']))
       die("ERREUR : tous les champs doivent être
       ✂ remplis.");
  if (preg_match("/^\d+(\.\d+)?$/",$_REQUEST['prix']) == false)
    die("ERREUR : prix non valide.");

  $liendb = mysql_connect($bddserver, $bddlogin,
  ✂ $bddpassword);
  mysql_select_db ($bdd);

  $sql = "INSERT INTO $table_produit (reference, nom,
  ✂ prix, description)".
         " VALUES (".
         "'".$_REQUEST['reference']."',".
         "'".$_REQUEST['nom']."',".
         "'".$_REQUEST['prix']."',".
         "'".$_REQUEST['description']."')";
  mysql_query ($sql);

  mysql_close($liendb);

}

include("haut.inc.php");

?>
```

```
<p>:: ajouter un produit au catalogue de la boutique</p>

<form action="<?php echo $_SERVER['PHP_SELF']; ?>"
✂ method="post">
<input type="hidden" name="enregistre" value="oui" />

<table>
<tr>
 <td>référence</td>
 <td><input type="text" name="reference" /></td>
 <td rowspan="3" valign="top">description</td>
 <td rowspan="3"><textarea name="description"
 ✂ rows="8"></textarea></td>
</tr>
<tr>
 <td>nom</td>
 <td><input type="text" name="nom" /></td>
</tr>
<tr>
 <td>prix</td>
 <td><input type="text" name="prix" /></td>
</tr>
</table>

<br/>

<input type="submit" value="enregistrer le produit" />

</form>

<hr>

<p>:: les produits de la boutique</p>

<table width="98%" align="center" border="1">
 <tr>
  <td class="intitule">id</td>
  <td class="intitule">référence</td>
  <td class="intitule">nom</td>
  <td class="intitule">prix  </td>
 </tr>

<?php

$liendb = mysql_connect($bddserver, $bddlogin,
✂ $bddpassword);
mysql_select_db ($bdd);

$sql = "SELECT * FROM $table_produit";
$resultat = mysql_query ($sql);

if (mysql_num_rows($resultat) > 0) {
```

```
    while ($produit = mysql_fetch_array ($resultat)) {
      $id = $produit['idproduit'];
      $reference = $produit['reference'];
      $nom = $produit['nom'];
      $prix = $produit['prix'];
      echo "<tr>";
      echo " <td>$id</td>";
      echo " <td>$reference</td>";
      echo " <td>$nom</td>";
      echo " <td>$prix</td>";
      echo "</tr>";
    }
  }
  else
  {
    echo "<tr><td colspan='4' align='center'>aucun
    ✂ produit</td></tr>";
  }

  echo "</table>";

  mysql_close($liendb);

  include("bas.inc.php");

  ?>
```

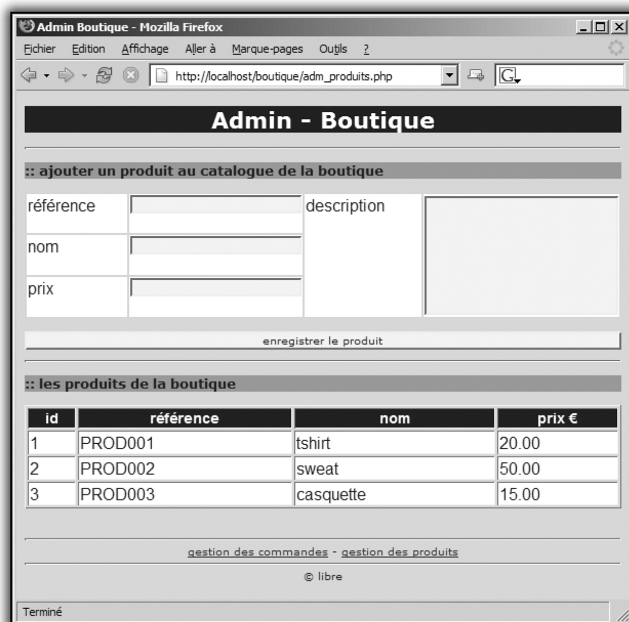


Figure 15.3 : *Page d'administration de la boutique*

Listing 15-8 : Le script adm_commandes.php

```
<?php

include("variables.inc.php");
include("identification.inc.php");
include("haut.inc.php");

?>

<p>:: les commandes</p>

<table>
 <tr>
  <td class="intitule">id</td>
  <td class="intitule">contenu</td>
  <td class="intitule">client</td>
  <td class="intitule">montant  </td>
  <td class="intitule">date</td>
 </tr>

<?php

$liendb = mysql_connect($bddserver, $bddlogin,
✂ $bddpassword);
mysql_select_db ($bdd);

$sql = "SELECT * FROM $table_commande";
$resultat = mysql_query ($sql);

if (mysql_num_rows($resultat) > 0) {

  while ($produit = mysql_fetch_array ($resultat)) {
    $id = $produit['idcommande'];
    $produits =
    ✂ str_replace(",","<br/>",$produit['produits']);
    $client = $produit['prenom']."
    ✂ ".$produit['nom']."<br/>".nl2br($produit['adresse']);
    $montant = $produit['montant'];
    $date = $produit['date'];
    echo "<tr>";
    echo "<td>$id</td>";
    echo "<td>$produits</td>";
    echo "<td>$client</td>";
    echo "<td>$montant</td>";
    echo "<td>$date</td>";
    echo "</tr>";
  }
}
else echo "<tr><td colspan='5'>aucune commande</td></tr>";
```

```
echo "</table>";

mysql_close($liendb);

include("bas.inc.php");

?>
```

Listing 15-9 : Le script haut.inc.php

```
<html>
<head>

<title>Admin Boutique</title>

<style type="text/css">
<!--

*
{ font-family:verdana; font-size:10px;}

BODY
{ background:#D1DAE3; color:#01291A; }

TABLE
{width:100%; }

H1
{ background: #091F35; color: white; font-size:22px;
✂ font-weight: bold; }

P
{ background:#8F9BB7; color:#091F35; font-size:13px;
✂ font-weight:bold; text-align:left; }

TD
{ background:white; color:#091F35; font-family:arial;
✂ font-size:16px; font-weight:normal; vertical-align:top; }

TD.intitule
{ background:#091F35; color:white; font-size:14px;
✂ font-weight:bold; text-align:center; }

INPUT
{ background:#FCF0E9; color:#01291A; width:100%; }

TEXTAREA
{ background:#FCF0E9; color:#01291A; width:100%; }
```

```
SELECT
{ background:#FCF0E9; color:#01291A; }

HR
{ color: #091F35; }

A
{ color: #6C111E; }

A:hover
{ background:#6C111E; color:white; font-weight:bold;}

-->
</style>

</head>
<body>

<center>

<h1>Admin - Boutique</h1>

<hr/>
```

Listing 15-10 : Le script bas.inc.php

```
<br/><br/><hr/>

<a href="adm_commandes.php">gestion des commandes</a> -
<a href="adm_produits.php">gestion des produits</a>

<hr/>

© libre

</center>

</body>
</html>
```

Listing 15-11 : Le script identification.inc.php

```
<?php

if ($_SERVER['PHP_AUTH_USER']!= "essai" ||
    $_SERVER['PHP_AUTH_PW']!="essai") {

  header("status: 401 Unauthorized");
  header("HTTP/1.0 401 Unauthorized");
  header("WWW-authenticate: basic realm=\"acces
  ✂ securise\"");
```

```
  print("Echec");
  exit(0);

}

?>
```

Listing 15-12 : variables.inc.php

```
<?php

$bddserver       = "localhost";
$bddlogin        = "root";
$bddpassword     = "";
$bdd             = "test";
$table_commande = "commande";
$table_produit  = "produit";
$url             = "http://localhost/boutique";

?>
```

Vous constatez la présence, dans le script *adm_produits.php*, de la ligne suivante :

```
<form action="<?php echo $PHP_SELF; ?>" method="post">
```

La variable `$_SERVER['PHP_SELF']` est une variable interne de PHP, qui contient à tout moment l'URL du script appelé.



Variables prédéfinies

PHP dispose, par défaut, d'un certain nombre de variables prédéfinies. Celles-ci contiennent des informations sur le serveur web, la page appelée, l'internaute, etc.



Une liste exhaustive de ces variables est fournie dans les annexes.

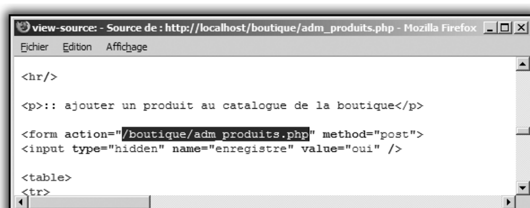


Figure 15.4 : *Contenu de la variable PHP_SELF*

Il peut être très intéressant de passer par cette variable quand vous souhaitez faire référence au script dans lequel vous vous trouvez. De cette manière, vous limitez le nombre de mises à jour si vous devez renommer le script.



`$_SERVER['PHP_SELF']` **et** `include()`

Si un fichier *f1.php* (**http://localhost/f1.php**) inclut un fichier *f2.php*, la variable `$_SERVER['PHP_SELF']` contiendra `/f1.php` dans `f1.php` ET dans `f2.php`. `$_SERVER['PHP_SELF']` contient l'URL appelée, et non le chemin du fichier dans lequel on se trouve !

Dans le script *adm_commandes.php*, vous remarquez l'utilisation des fonctions `nl2br()` et `str_replace()`.

- `nl2br()` : lorsque vous enregistrez dans une table, une donnée multiligne provenant d'un `textarea`, cette donnée est stockée avec des retours à la ligne (codés `\n`). Pour l'afficher dans une page web, il est donc nécessaire d'utiliser la fonction `nl2br()` pour convertir les retours à la ligne texte (`\n`) en retours à la ligne HTML (`<br/>`).
- `str_replace()` : permet de remplacer un texte par un autre dans une chaîne de caractères. Si vous souhaitez remplacer la chaîne `"toto"` par la chaîne `"titi"` dans la chaîne `$ch` et stocker le résultat dans la chaîne `$ch2`, il suffit d'écrire `$ch2 = str_replace("toto","titi",$ch);`.



Une description plus approfondie de cette fonction est fournie dans le chapitre « Fonctions PHP ». Vous y découvrirez notamment les possibilités impressionnantes qu'elle offre lorsqu'elle est utilisée en adéquation avec les expressions régulières.

Le front-office

Le front office correspond dans cet exemple à la zone du site qui permettra à l'internaute de sélectionner ses achats et de passer sa commande. Commencez par réaliser la page d'accueil et la page catalogue :

Listing 15-13 : Le fichier index.html

```
<html>
 <head>
  <title>Boutique FoxShop</title>
  <link href="look.css" rel="stylesheet" type="text/css" />
 </head>
<body>

<div class='titre'><a href='boutique.php'>Boutique
✂ <i>FoxSHOP</i></a></div>

<div class='bienvenue'>Welcome</div>

</body>
</html>
```

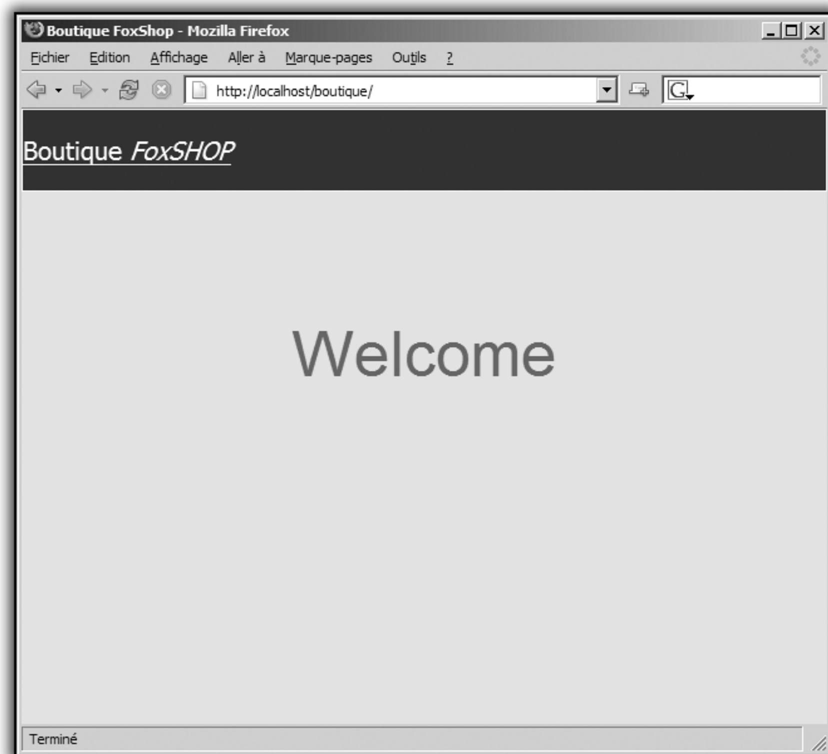


Figure 15.5 : *Page d'accueil de la boutique*

A priori, rien à signaler, si ce n'est la gestion des styles (CSS). Comme la page d'accueil est de type HTML, vous ne pouvez recourir à la

technique d'inclusion et utiliser un fichier *haut.inc.php* pour inclure les caractéristiques visuelles de la boutique. Heureusement, une autre technique permet d'isoler les CSS en un endroit unique. Les propriétés CSS peuvent être incluses dans un fichier extérieur qui sera lié à la page web de la manière suivante :

```
<link href="look.css" rel="stylesheet" type="text/css" />
```

Dans ce cas, ce fichier s'appelle *look.css*, il se situe au même niveau que le script *index.php* et contient les instructions suivantes :

```
BODY
{background: #025053;}

TD
{background: #C2DADB; color: #022324; font-family:
✂ georgia;
 font-weight: bold; font-size: 14px; letter-spacing: 1px;}

A
{color: #EB8814; font-family: georgia; font-weight: bold;
 font-size: 14px; letter-spacing: 1px;}

.titre
{color: #047F84; font-family: georgia; font-weight: bold;
 font-size: 24px; letter-spacing: 1px;}

.reference
{color: #047F84; font-family: arial; font-weight: bold;
 font-size: 12px; letter-spacing: 1px;}

.description
{background: #DAE9EA; color: #047F84; font-family:
✂ verdana;
 font-weight: normal; font-size: 12px; letter-spacing: 1px;}
```

Passez maintenant au script *boutique.php*, qui permet d'accéder à l'ensemble du catalogue :

Listing 15-14 : boutique.php

```
<?php

include("variables.inc.php");

if (!isset($_REQUEST['id'])) $id = 1;
else $id = $_REQUEST['id'];

?>

<html>
```

```
 <head>
  <title>Boutique FoxShop - catalogue</title>
  <link href="look.css" rel="stylesheet" type="text/css" />
 </head>
<body>

<div class='titre'><a href='boutique.php'>Boutique
✂ <i>FoxSHOP</i></a></div>

<table class='catalogue'>
<tr>
 <td class='liste'>
  <div class='tdTitre'>Nos produits</div>

<?php
$liendb = mysql_connect($bddserver, $bddlogin,
✂ $bddpassword);
mysql_select_db ($bdd);
$sql = "SELECT * FROM $table_produit";
$resultat = mysql_query ($sql);
while ($produit = mysql_fetch_array ($resultat)) {
  print("  - ");
  print("<a href=".$_SERVER['PHP_SELF']."?id=".$produit
  ✂ ['idproduit'].">".
        $produit['nom']."</a>");
  print("<br/>");
}
?>

 </td>
 <td class='detail'>

<?php
$sql = "SELECT * FROM $table_produit WHERE idproduit =
✂ '$id'";
$resultat = mysql_query ($sql);
$produit = mysql_fetch_array ($resultat);
print("<div class='tdTitre'>".$produit['nom'].
      " [ref#".$produit['reference']."]</div>");
?>

<div class='description'>

<?php
print(nl2br($produit['description'])."<br/><br/>");
print($produit['prix']."  <br/><br/>");
mysql_close($liendb);
?>

<form action="ajout_caddie.php" method="post">
 <input type="hidden" name="id" value="<?php echo $id; ?>" />
```

```
 <input type="submit" value="ajouter au panier" />
</form>

</div>

 </td>
</tr>
</table>

</body>
</html>
```

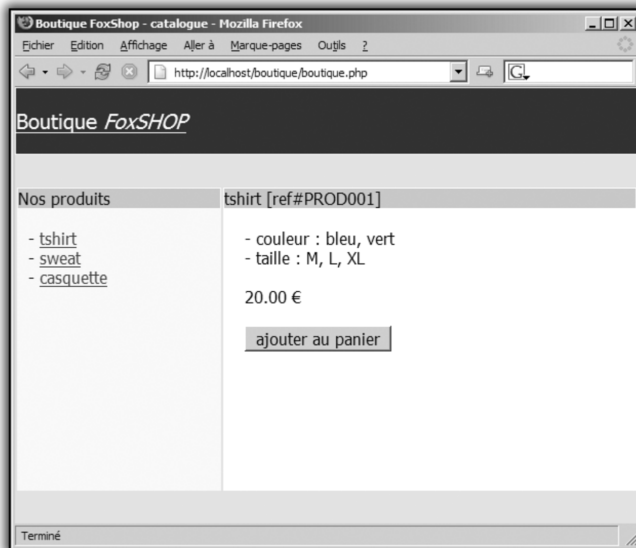


Figure 15.6 : *Catalogue de la boutique*

Quand aucun produit n'est spécifié, vous prenez la valeur par défaut `1` :

```
if (!isset($_REQUEST['id'])) $id = 1;
else $id = $_REQUEST['id'];
```

C'est par exemple le cas quand vous venez de la page d'accueil.

En revanche, lorsque vous cliquez sur le nom d'un produit, vous passez le paramètre `id` au script et lui permettez ainsi d'aller chercher les caractéristiques du produit associé. Vous pouvez donc dire que le catalogue de la boutique est entièrement dynamique.

Le bouton `"ajouter au panier"` va permettre de passer la commande en basculant sur *ajout_caddie.php*.

Venons-en justement au cœur du problème : la gestion du panier. L'idée est de construire un cookie `monpanier` qui contiendra les `id` de tous les produits qui y ont été ajoutés. Pour faire simple, vous choisirez le format de stockage `idprod1,idprod2,idprod3`, etc.

Le fichier *ajout_caddie.php* se contente donc de concaténer la chaîne `,idprodN` à la fin du cookie existant (s'il n'existe pas, il s'initialise). Une fois la manipulation sur le cookie réalisée, vous êtes redirigé sur la fiche du produit que vous venez d'acheter.

Le code du script *ajout_caddie.php* devient maintenant évident :

Listing 15-15 : Le script ajout_caddie.php

```
<?php
include("variables.inc.php");
setcookie("monpanier",$_COOKIE['monpanier'].",".$_REQUEST
✂ ['id'],time()+86400);
header("Location: $url/boutique.php?id=".$_REQUEST['id']);
?>
```

Vous disposez donc d'un panier et il devient nécessaire d'y faire référence dans la boutique.

Apportez quelques modifications à *boutique.php* de manière que les éléments suivants soient affichés (lorsque le panier existe) :

- le message `"Votre panier contient N articles"` ;
- un bouton pour valider la commande ;
- le contenu du panier pour rendre compte, *de visu*, du contenu du cookie.

Listing 15-16 : Vous disposez maintenant d'un panier dans la boutique

```
<?php

include("variables.inc.php");

if (!isset($_REQUEST['id'])) $id = 1;
else $id = $_REQUEST['id'];

?>

<html>
 <head>
  <title>Boutique FoxShop - catalogue</title>
  <link href="look.css" rel="stylesheet" type="text/
  ✂ css" />
```

```
 </head>
<body>

<div class='titre'><a href='boutique.php'>Boutique
✂ <i>FoxSHOP</i></a></div>

<table class='catalogue'>
<tr>
 <td class='liste'>
  <div class='tdTitre'>Nos produits</div>

<?php
$liendb = mysql_connect($bddserver, $bddlogin,
✂ $bddpassword);
mysql_select_db ($bdd);
$sql = "SELECT * FROM $table_produit";
$resultat = mysql_query ($sql);
while ($produit = mysql_fetch_array ($resultat)) {
  print("  - ");
  print("<a href=".$_SERVER['PHP_SELF']."?id="
  ✂ .$produit['idproduit'].">".
        $produit['nom']."</a>");
  print("<br/>");
}
?>

 </td>
 <td class='detail'>

<?php
$sql = "SELECT * FROM $table_produit WHERE idproduit
✂ = '$id'";
$resultat = mysql_query ($sql);
$produit = mysql_fetch_array ($resultat);
print("<div class='tdTitre'>".$produit['nom'].
      " [ref#".$produit['reference']."]</div>");
?>

<div class='description'>

<?php
print(nl2br($produit['description'])."<br/><br/>");
print($produit['prix']."  <br/><br/>");
mysql_close($liendb);
?>

<form action="ajout_caddie.php" method="post">
 <input type="hidden" name="id" value="<?php echo
 ✂ $id; ?>" />
 <input type="submit" value="ajouter au panier" />
</form>
```

```
<?php

if (isset($_COOKIE['monpanier']))
{
 print("<div class='panier'>");
 $tab = split(",",$_COOKIE['monpanier']);
 $nb_prod = sizeof($tab) - 1;
 print("votre panier contient ".$nb_prod."
 ✂ produit(s)<br/>");
 print("<form action='voir_caddie.php'
 ✂ method='post'>");
 print("<input type='submit' value='valider la
 ✂ commande'/></form>");
 print("cookie = {".$_COOKIE['monpanier']."}");
 print("</div>");
}

?>

</div>

 </td>
</tr>
</table>

</body>
</html>
```

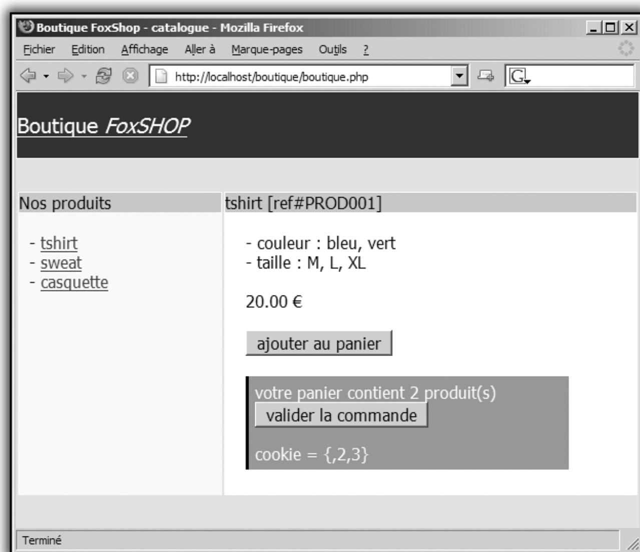


Figure 15.7 : *Vous avez acheté un sweater (id=2) et une casquette (id=3)*

Vous utilisez pour trouver le nombre d'éléments dans le panier la fonction `split()`. Utilisée sur la chaîne `",2,3"`, la fonction `split(",",",2,3")` retourne un tableau de trois éléments avec un premier élément vide. De ce fait, vous êtes obligé de décrémenter de `1` la taille du tableau pour trouver le nombre exact de produits :

```
$nb_prod = sizeof($tab) - 1;
```

Passez maintenant à la validation de la commande. Vous avez deux scripts à écrire :

- Le script *voir_caddie.php* liste les produits présents dans le panier et vous permet d'enregistrer vos informations client.
- Le script *enregistre_commande.php*, appelé par *voir_caddie.php*, enregistre la commande dans la base et supprime le caddie.

Listing 15-17 : Le script voir_caddie.php (voir Figure 12.8)

```
<?php
include("variables.inc.php");
?>

<html>
 <head>
  <title>Boutique FoxShop - validation
  ✂ commande</title>
  <link href="look.css" rel="stylesheet" type="text/
  ✂ css" />
 </head>
<body>

<div class='titre'><a href='boutique.php'>Boutique
✂ <i>FoxSHOP</i></a></div>

<div class='caddie'>

<?php
$montant = 0;
$listeproduits = " ";
$_COOKIE['monpanier'][0] = ' ';
$liendb = mysql_connect($bddserver, $bddlogin,
✂ $bddpassword);
mysql_select_db ($bdd);
$sql = "SELECT * FROM $table_produit ".
       "WHERE idproduit IN
       ✂ (".$_COOKIE['monpanier'].")";
$resultat = mysql_query ($sql);
print("<table width='100%'>");
while ($prod = mysql_fetch_array ($resultat)) {
  print("<tr>");
```

```
    print("<td class='prod'>[".$prod['reference']."]
    ✂ ".$prod['nom']."</td>");
    print("<td class='montant'>".$prod['prix']."
    ✂  </td></tr>");
    $montant += $prod['prix'];
    $listeproduits .= ',' .  $prod['reference'];
  }
  $listeproduits[0] = ' ';
  // frais de port
  $montant += 5;
  print("<tr><td class='total'>MONTANT + PORT</td>");
  print("<td class='total'>$montant  </td></tr>");
  print("</table>");
  mysql_close($liendb);
  ?>

  <form action="enregistre_commande.php" method="post">

   <input type="hidden" name="montant" value="<?php
   ✂ echo $montant; ?>">
   <input type="hidden" name="listeproduits"
         value="<?php echo $listeproduits; ?>">

   <label>nom</label><br/><input type="text" name="nom"
   ✂ /><br/>
   <label>prénom</label><br/><input type="text"
   ✂ name="prenom" /><br/>
   <label>adresse</label><br/><input type="text"
   ✂ name="adresse" /><br/>
   <label>code postal</label><br/><input type="text"
   ✂ name="cp" /><br/>
   <label>ville</label><br/><input type="text"
   ✂ name="ville" /><br/>
   <input type="submit"  value="enregistrer ma
   ✂ commande" />

  </form>

  </div>

  </body>
  </html>
```

(voir Figure 12.8)

Pour ne pas être gêné par les virgules initiales dans `$_COOKIE['monpanier']` et `$listeproduits`, remplacez la première lettre par un caractère blanc :

```
$monpanier[0] = ' ';
```

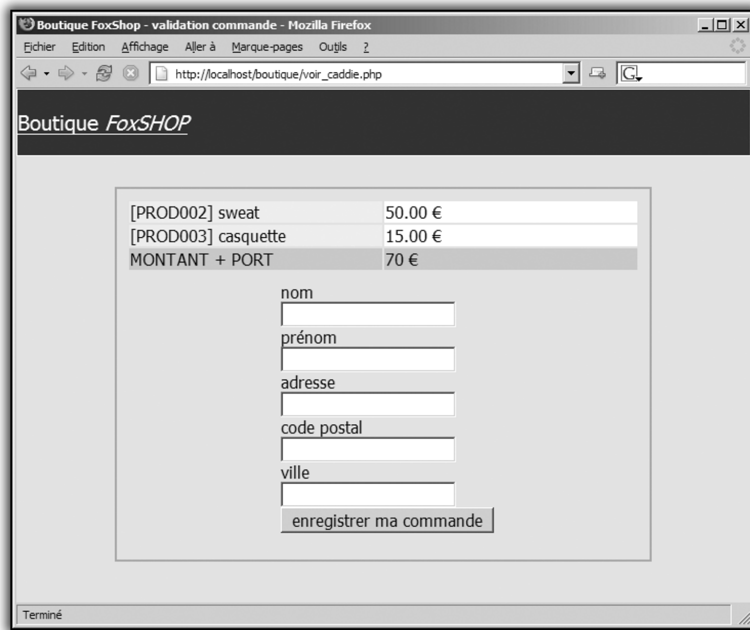


Figure 15.8 : *Visualisation du caddie*

Vous avez vu, en effet, qu'une chaîne de caractères pouvait être considérée comme un tableau de caractères. Il est donc possible de modifier un à un les caractères en y accédant par un index.

Supposons maintenant que vous ayez, dans votre panier, les produits `2` et `3`. Pour les sélectionner dans la base, la solution la plus évidente est la requête SQL suivante :

```
SELECT * FROM $table_produit WHERE idproduit = '2' OR
idproduit = '3'
```

Dans le cas présent, vous allez tirer parti d'une autre syntaxe, plus compacte, qui permet d'aboutir au même résultat.

La ligne suivante :

```
SELECT * FROM $table_produit WHERE idproduit IN (2,3)
```

… est équivalente à la requête précédente et signifie : « sélectionner tous les produits dont l'`idproduit` est contenu sur la liste suivante (2,3) ».

Ce procédé comporte cependant un sérieux défaut : si vous ajoutez plusieurs fois le même produit au panier, celui-ci n'apparaît qu'une

seule fois dans votre récapitulatif. Il faut donc préciser pour chaque produit sa quantité dans le panier. La fonction `array_count_values()`, qui dénombre le nombre d'occurrences d'une donnée dans un tableau, peut vous être utile.

Listing 15-18 : Le script voir_caddie.php

```
<?php
include("variables.inc.php");
?>

<html>
 <head>
  <title>Boutique FoxShop - validation commande</title>
  <link href="look.css" rel="stylesheet" type="text/css" />
 </head>
<body>

<div class='titre'><a href='boutique.php'>Boutique
✂ <i>FoxSHOP</i></a></div>

<div class='caddie'>

<?php
$montant = 0;
$listeproduits = " ";
$_COOKIE['monpanier'][0] = ' ';
$liendb = mysql_connect($bddserver, $bddlogin,
✂ $bddpassword);
mysql_select_db ($bdd);
$sql = "SELECT * FROM $table_produit ".
       "WHERE idproduit IN (".$_COOKIE['monpanier'].")";
$resultat = mysql_query ($sql);
print("<table width='100%'>");
$tab = array_count_values(split(",",$_COOKIE['monpanier']));
while ($prod = mysql_fetch_array ($resultat)) {
  print("<tr><td class='prod'>");
  print("[".$prod['reference']."] ".$prod['nom']);
  print(" (x".$tab[$prod['idproduit']].")");
  print("</td><td class='montant'>");
  print($prod['prix']."  ");
  print("</td></tr>");
  $montant += $prod['prix']*$tab[$prod['idproduit']];
  $listeproduits .= ',' .  $prod['reference'];
}
$listeproduits[0] = ' ';
// frais de port
$montant += 5;
print("<tr><td class='total'>MONTANT + PORT</td>");
print("<td class='total'>$montant  </td></tr>");
print("</table>");
```

```
mysql_close($liendb);
?>

<form action="enregistre_commande.php" method="post">

 <input type="hidden" name="montant" value="<?php echo
 ✂ $montant; ?>">
 <input type="hidden" name="listeproduits"
        value="<?php echo $listeproduits; ?>">

 <label>nom</label><br/><input type="text" name="nom" /><br/>
 <label>prénom</label><br/><input type="text" name="prenom"
 ✂ /><br/>
 <label>adresse</label><br/><input type="text"
 ✂ name="adresse" /><br/>
 <label>code postal</label><br/><input type="text"
 ✂ name="cp" /><br/>
 <label>ville</label><br/><input type="text" name="ville"
 ✂ /><br/>
 <input type="submit"  value="enregistrer ma commande" />

</form>

</div>

</body>
</html>
```

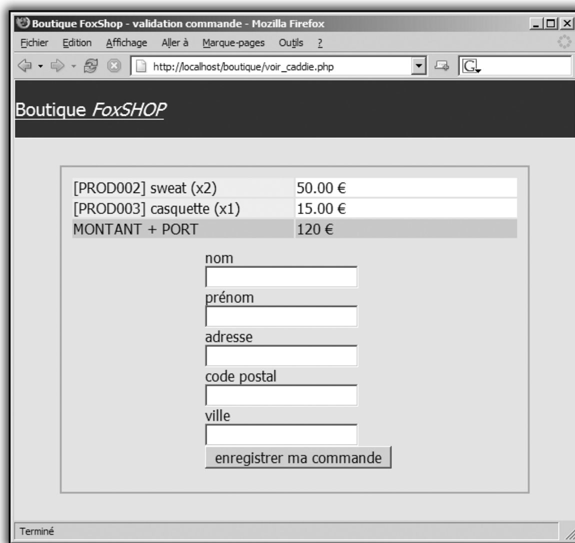


Figure 15.9 :
Le même produit est désormais listé deux fois

Passez maintenant à l'enregistrement de la commande :

Listing 15-19 : Le script enregistre_commande.php

```
<?php

include("variables.inc.php");

if (empty($_REQUEST['nom']) || empty($_REQUEST['prenom']) ||
    empty($_REQUEST['adresse']) || empty($_REQUEST['cp']) ||
    empty($_REQUEST['ville']))
    die("ERREUR : tous les champs doivent être remplis.");

$liendb = mysql_connect($bddserver, $bddlogin,
✂ $bddpassword);
mysql_select_db ($bdd);

$date = date("Y-m-d G:i:s");

$_COOKIE['monpanier'][0] = ' ';

$tab_prod = split(",",$_COOKIE['monpanier']);

$i = 0;
while ($id = $tab_prod[$i]) {
  $sql = "SELECT * FROM $table_produit WHERE idproduit =
  ✂ '$id'";
  $resultat = mysql_query ($sql);
  $produit = mysql_fetch_array ($resultat);
  $montant += $produit['prix'];
  $listeproduits .= ',' .  $produit['reference'];
  $i++;
}
$listeproduits[0] = ' ';
$montant += 5;
$date = date("Y-m-d G:i:s");
$sql = "INSERT INTO $table_commande (produits, montant,
✂ nom, prenom, adresse, date) VALUES ('"
✂ .$_REQUEST['listeproduits']."', '".$_REQUEST['montant'
✂ ]."', '".$_REQUEST['nom']."', '".$_REQUEST['prenom']."',
'".$_REQUEST['adresse']."\n".$_REQUEST['cp']."\n"
✂ .$_REQUEST['ville']."', '$date')";

mysql_query ($sql);

mysql_close($liendb);

setcookie("monpanier","",time()-3600);

header("Location: $url/boutique.php");

?>
```

La suppression du cookie est réalisée avec l'instruction suivante :

```
setcookie("monpanier","",time()-3600);
```



Choix du procédé

Nous préférons, pour supprimer le cookie, cette dernière syntaxe à la syntaxe `setcookie("monpanier")`, car elle est plus radicale. L'idée ici est de mettre une date d'expiration dans le passé pour s'assurer que le navigateur efface bien le cookie.

La mini-boutique est donc achevée, les commandes sont bien enregistrées dans la base.

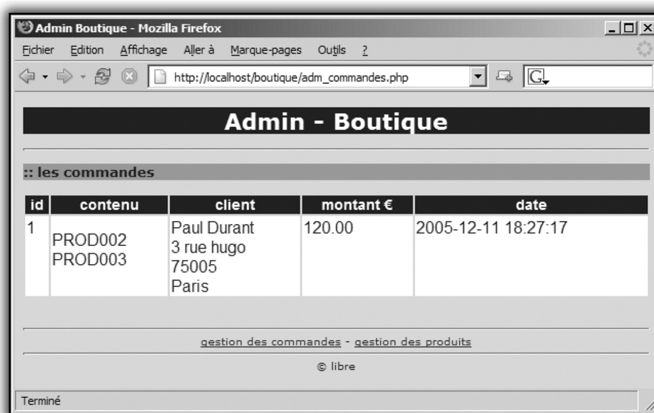


Figure 15.10 : Administration des commandes

Est-ce satisfaisant ? Certainement pas. Imaginez ce que pourrait entraîner le fait de taper l'URL suivante : `http://localhost/boutique/enregistre_commande.php?montant=10&liste produits=PROD001,PROD002,PROD002&nom=Daunou&prenom =Sophie&adresse=41+rue+voltaire&cp=75002&ville=Paris`.

Cette commande permet de valider une commande de trois produits pour un montant de 10 euros ! L'exemple montre bien que, dès que vous réalisez le moindre applicatif à vocation publique, il est impératif, à un moment donné, de se mettre dans l'état d'esprit d'une personne aux intentions douteuses.

L'erreur ici est de passer directement en paramètre le montant de la commande. Le fait d'avoir la liste des produits permet cependant

d'éviter cela. C'est le script *enregistre_commande.php* qui doit calculer lui-même le montant de la commande. Comme vous allez vous baser sur le contenu du cookie, vous pouvez aussi vous débarrasser du paramètre `listeproduits`. Les paramètres `hidden`, du script *voir_caddie.php*, peuvent donc être supprimés :

Listing 15-20 : Version sécurisée de enregistre_commande.php

```
<?php

include("variables.inc.php");

if (empty($_REQUEST['nom']) || empty($_REQUEST['prenom']) ||
    empty($_REQUEST['adresse']) || empty($_REQUEST['cp']) ||
    empty($_REQUEST['ville']))
    die("ERREUR : tous les champs doivent être remplis.");

$liendb = mysql_connect($bddserver, $bddlogin, $bddpassword);
mysql_select_db ($bdd);

$date = date("Y-m-d G:i:s");

$_COOKIE['monpanier'][0] = ' ';

$tab_prod = split(",",$_COOKIE['monpanier']);

$i = 0;
while ($id = $tab_prod[$i]) {
  $sql = "SELECT * FROM $table_produit WHERE idproduit = '$id'";
  $resultat = mysql_query ($sql);
  $produit = mysql_fetch_array ($resultat);
  $montant += $produit['prix'];
  $listeproduits .= ',' .  $produit['reference'];
  $i++;
}

$listeproduits[0] = ' ';

$montant += 5;

$date = date("Y-m-d G:i:s");

$sql = "INSERT INTO $table_commande (produits, montant,
✂ nom, prenom, adresse, date) VALUES ('$listeproduits',
✂ '$montant', '".$_REQUEST['nom']."', '"
✂ .$_REQUEST['prenom']."', '".$_REQUEST['adresse']."\n"
✂ .$_REQUEST['cp']."\n".$_REQUEST['ville']."', '$date')";

mysql_query ($sql);
```

```
mysql_close($liendb);

setcookie("monpanier","",time()-3600);

header("Location: $url/boutique.php");

?>
```

Dans le cadre de cette application, les cookies vont servir également à stocker le profil client et lui préremplir son formulaire d'identification en cas de nouvel achat.

L'idée est donc de créer un nouveau cookie, `$_COOKIE['monprofil']`, qui contient toutes les informations des clients. La création doit se faire lors de l'enregistrement de la commande dans *enregistre_commande.php*. Vous allez choisir la norme `nom1=valeur1;;nom2=valeur2;;…` pour stocker les données et ajouter la ligne suivante dans *enregistre_commande.php* :

```
setcookie("monprofil","nom=".$_REQUEST['nom'].";;prenom="
✂ .$_REQUEST['prenom'].";;adresse=".$_REQUEST['adresse']
✂ .";;cp=".$_REQUEST['cp'].";;ville=".$_REQUEST['ville']
✂ ."",time()+604800);
```



Norme pour le stockage de données

Il est courant en PHP de stocker les données dans son propre format. Veillez alors à ne pas utiliser de caractères trop répandus comme caractères de séparation afin d'éviter tout risque de conflit avec l'une des valeurs. Dans l'exemple suivant, si vous choisissez la virgule comme caractère de séparation, vous serez bien ennuyé pour différencier la virgule de séparation de la virgule présente dans l'adresse : `adresse=9, rue Jean-Jaurès,CP=75015,ville=Paris`.

La deuxième étape consiste à préremplir le formulaire. La seule difficulté est de récupérer chaque caractéristique du profil. Il faut utiliser une première fois la fonction `split()` avec deux points-virgules (`";;"`) comme premier paramètre et une deuxième fois avec le signe égal (=).

Listing 15-21 : Le formulaire est désormais pré rempli

```
<?php
include("variables.inc.php");
?>

<html>
```

```
 <head>
  <title>Boutique FoxShop - validation commande</title>
  <link href="look.css" rel="stylesheet" type="text/css" />
 </head>
<body>

<div class='titre'><a href='boutique.php'>Boutique
✂ <i>FoxSHOP</i></a></div>

<div class='caddie'>

<?php
$montant = 0;
$listeproduits = " ";
$_COOKIE['monpanier'][0] = ' ';
$liendb = mysql_connect($bddserver, $bddlogin, $bddpassword);
mysql_select_db ($bdd);
$sql = "SELECT * FROM $table_produit ".
       "WHERE idproduit IN (".$_COOKIE['monpanier'].")";
$resultat = mysql_query ($sql);
print("<table width='100%'>");
$tab = array_count_values(split(",",$_COOKIE['monpanier']));
while ($prod = mysql_fetch_array ($resultat)) {
  print("<tr><td class='produit'>");
  print("[".$prod['reference']."] ".$prod['nom']);
  print(" (x".$tab[$prod['idproduit']].")");
  print("</td><td class='montant'>");
  print($prod['prix']."  ");
  print("</td></tr>");
  $montant += $prod['prix']*$tab[$prod['idproduit']];
  $listeproduits .= ','.$prod['reference'];
}
$listeproduits[0] = ' ';
// frais de port
$montant += 5;
print("<tr><td class='total'>MONTANT + PORT</td>");
print("<td class='total'>$montant  </td></tr>");
print("</table>");
mysql_close($liendb);
?>

<form action="enregistre_commande.php" method="post">

 <input type="hidden" name="montant" value="<?php echo
 ✂ $montant; ?>">
 <input type="hidden" name="listeproduits"
        value="<?php echo $listeproduits; ?>">

<?php
if (!empty($_COOKIE['monprofil'])) {
  $tab_tmp = split(";;",$_COOKIE['monprofil']);
```

```
  $i = 0;
  while ($tab_tmp[$i]) {
     list ($nom,$val) = split("=",$tab_tmp[$i]);
     $tab_profil[$nom] = $val;
     $i++;
  }
}
?>

 <label>nom</label><br/>
 <input type="text" name="nom" value="<?php echo
 ✂ $tab_profil['nom']?>" />
 <br/><label>prénom</label><br/>
 <input type="text" name="prenom"
        value="<?php echo $tab_profil['prenom']?>" />
 <br/><label>adresse</label><br/>
 <input type="text" name="adresse"
        value="<?php echo $tab_profil['adresse']?>" />
 <br/><label>code postal</label><br/>
 <input type="text" name="cp"
        value="<?php echo $tab_profil['cp']?>" />
 <br/><label>ville</label><br/>
 <input type="text" name="ville"
        value="<?php echo $tab_profil['ville']?>" />
 <br/><input type="submit"  value="enregistrer ma commande" />

</form>

</div>

</body>
</html>
```

Plutôt qu'un deuxième tableau temporaire pour recueillir les données issues du deuxième `split()`, utilisez la fonction `list` qui vous permet de placer directement les données dans deux variables.

Si l'internaute n'est pas encore client, le tableau `$tab_profil` est vide. De ce fait, afficher `$tab_profil['nom']` n'est pas gênant, car cela équivaut à ne rien afficher. Si, par contre, il est déjà client, `$tab_profil['nom']` contiendra le nom qu'il avait spécifié lors de son dernier achat. Quand c'est possible, comme ici, il est intéressant de ne pas multiplier les cas (avec des `if`…) et d'essayer d'être le plus générique possible.

15.2. Les sessions

Vous venez de voir que les cookies permettent de transmettre de l'information d'une page à l'autre. Certains inconvénients peuvent cependant être notés :

- l'obligation de les gérer en haut du script avant que des données ne soient affichées ;
- le format assez primaire de stockage de l'information (une chaîne de caractères).

Pour répondre à ces limitations, le langage PHP vous propose d'utiliser un système de session. À n'importe quel endroit de votre script, il est possible de définir certaines variables en tant que variable de session. Une fois enregistrées, ces variables sont propagées d'une page à l'autre. Elles deviennent schématiquement des variables globales à l'ensemble du site, tout en restant associées à un visiteur donné. Si vous modifiez le contenu d'une variable de session dans un script *A*, vous récupérez le contenu modifié dans un script *B*. Il est important de signaler que le nombre de variables de session n'est pas limité : vous pouvez en enregistrer autant que nécessaire.

Le mode de fonctionnement des sessions est extrêmement simple. Il s'appuie sur :

- la fonction `session_start()` qui indique au script que vous souhaitez récupérer les variables de session.
- la variable super-globale `$_SESSION` qui contient l'ensemble des variables de session.

  ```
  $_SESSION['x'] = 2;
  ```

Réalisez ce petit exemple, qui permet de savoir combien de fois une personne est venue sur une page :

Listing 15-22 : Exemple simple d'utilisation des sessions

```
<?php

session_start();

if (!isset($_SESSION['visite'])) {
  echo "première visite";
  $_SESSION['visite'] = 1;
}
else {
```

```
    $_SESSION['visite']++;
    echo "vous avez visité cette page ".$_SESSION['visite'
    ✂ ]." fois";
}

?>
```

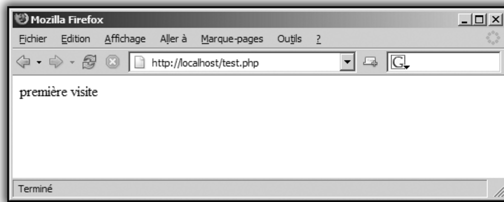


Figure 15.11 : *Message apparaissant en arrivant pour la première fois sur la page*

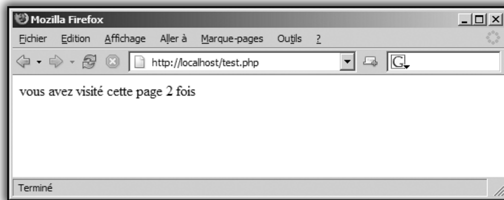


Figure 15.12 : *Message apparaissant lorsque la page est rafraîchie*

Analysons cet exemple…

Dans ce script, vous souhaitez faire usage de variables de session. Vous l'indiquez à PHP en exécutant la fonction `session_start()`. Votre seule contrainte est de faire appel à cette fonction avant le premier affichage du script.

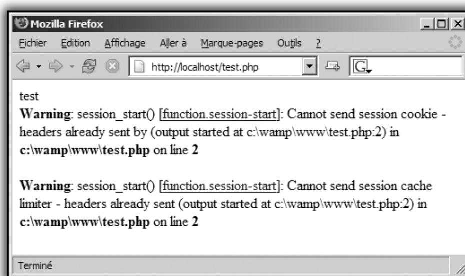


Figure 15.13 : *Erreur générée à la suite d'un affichage de la chaîne "test" avant l'appel à session_start()*

Vous testez ensuite si la variable $_SESSION['visite'] (qui va contenir le nombre de vos visites) existe. Si ce n'est pas le cas, vous affichez le message "première visite" et initialisez la variable de session "visite" à 1. À partir de ce moment, toute modification de sa valeur sera propagée d'une page à l'autre.

En rafraîchissant la page, vous vous retrouvez dans le cas où la variable $_SESSION['visite'] existe. Lors de chaque accès à cette page, vous faites augmenter le compteur de visites en incrémentant directement la variable $_SESSION['visite'].

Bien évidemment, si vous placez ce même code dans un autre script, la variable $_SESSION['visite'] sera aussi incrémentée. La variable de session est en effet partagée par tous les scripts d'un même site.

Un autre grand avantage des variables de session est de permettre d'enregistrer des variables de type complexe comme des tableaux. Vous pouvez voir l'extrême intérêt de cette fonctionnalité dans le cadre de votre applicatif. Plutôt que de stocker vos données en les séparant par des virgules (valeur1,valeur2, etc.) ou en utilisant une norme boiteuse (nom1=valeur1;;nom2=valeur2;;…), autant utiliser dans le premier cas un tableau scalaire, et, dans le deuxième cas, un tableau associatif.

Modifiez les fichiers *ajout_caddie.php*, *boutique.php*, *voir_caddie.php*, *enregistre_commande.php* afin d'utiliser des sessions plutôt que des cookies.

Commencez par l'ajout au panier :

Listing 15-23 : Le script ajout_caddie.php

```
<?php

include("variables.inc.php");

session_start();

if (!isset($_SESSION['monpanier'])) $_SESSION['monpanier']
✂ = array();

$_SESSION['monpanier'][] = $_REQUEST['id'];

header("Location: $url/boutique.php?id=".$_REQUEST['id']);

?>
```

Listing 15-24 : Le script boutique.php

```
<?php

include("variables.inc.php");
session_start();

if (!isset($_REQUEST['id'])) $id = 1;
else $id = $_REQUEST['id'];

?>

<html>
 <head>
  <title>Boutique FoxShop - catalogue</title>
  <link href="look.css" rel="stylesheet" type="text/css" />
 </head>
<body>

<div class='titre'><a href='boutique.php'>Boutique
✂ <i>FoxSHOP</i></a></div>

<table class='catalogue'>
<tr>
 <td class='liste'>
  <div class='tdTitre'>Nos produits</div>

<?php
$liendb = mysql_connect($bddserver, $bddlogin,
✂ $bddpassword);
mysql_select_db ($bdd);
$sql = "SELECT * FROM $table_produit";
$resultat = mysql_query ($sql);
while ($produit = mysql_fetch_array ($resultat)) {
  print("  - ");
  print("<a href=".$_SERVER['PHP_SELF']."?id=".$produit
  ✂ ['idproduit'].">". $produit['nom']."</a>");
  print("<br/>");
}
?>

 </td>
 <td class='detail'>

<?php
$sql = "SELECT * FROM $table_produit WHERE idproduit = '$id'";
$resultat = mysql_query ($sql);
$produit = mysql_fetch_array ($resultat);
print("<div class='tdTitre'>".$produit['nom'].
      " [ref#".$produit['reference']."]</div>");
?>
```

```
<div class='description'>

<?php
print(nl2br($produit['description'])."<br/><br/>");
print($produit['prix']."  <br/><br/>");
mysql_close($liendb);
?>

<form action="ajout_caddie.php" method="post">
 <input type="hidden" name="id" value="<?php echo $id; ?>" />
 <input type="submit" value="ajouter au panier" />
</form>

<?php

if (isset($_SESSION['monpanier']))
{
 print("<div class='panier'>");
 $nb_prod = count($_SESSION['monpanier']);
 print("votre panier contient ".$nb_prod." produit(s)<br/>");
 print("<form action='voir_caddie.php' method='post'>");
 print("<input type='submit' value='valider la commande'/>
 ✂ </form>");
 print("session = {".implode(",",$_SESSION['monpanier'])."}");
 print("</div>");
}

?>

</div>
 </td>
</tr>
</table>

</body>
</html>
```

Le passage aux sessions a globalement clarifié et simplifié le code. Les différents produits ajoutés au panier sont désormais stockés dans une variable de session nommée `$_SESSION['monpanier']`, de type tableau scalaire.

Passez maintenant à la validation et à l'enregistrement de la commande :

Listing 15-25 : Le script voir_caddie.php

```
 <?php

include("variables.inc.php");
```

```
session_start();

?>

<html>
 <head>
  <title>Boutique FoxShop - validation commande</title>
  <link href="look.css" rel="stylesheet" type="text/css" />
 </head>
<body>

<div class='titre'><a href='boutique.php'>Boutique
✂ <i>FoxSHOP</i></a></div>

<div class='caddie'>

<?php
$montant = 0;
$listeproduits = " ";
$liendb = mysql_connect($bddserver, $bddlogin, $bddpassword);
mysql_select_db ($bdd);
$sql = "SELECT * FROM $table_produit ".
       "WHERE idproduit IN (".implode(',',$_SESSION
       ✂ ['monpanier']).")";
$resultat = mysql_query ($sql);
print("<table width='100%'>");
$tab = array_count_values($_SESSION['monpanier']);
while ($prod = mysql_fetch_array ($resultat)) {
  print("<tr><td class='produit'>");
  print("[".$prod['reference']."] ".$prod['nom']);
  print(" (x".$tab[$prod['idproduit']].")");
  print("</td><td class='montant'>");
  print($prod['prix']."  ");
  print("</td></tr>");
  $montant += $prod['prix']*$tab[$prod['idproduit']];
  $listeproduits .= ','.$prod['reference'];
}
$listeproduits[0] = ' ';
// frais de port
$montant += 5;
print("<tr><td class='total'>MONTANT + PORT</td>");
print("<td class='total'>$montant  </td></tr>");
print("</table>");
mysql_close($liendb);
?>

<form action="enregistre_commande.php" method="post">

 <input type="hidden" name="montant" value="<?php echo
 ✂ $montant; ?>">
 <input type="hidden" name="listeproduits"
```

```
        value="<?php echo $listeproduits; ?>">

 <label>nom</label><br/>
 <input type="text" name="nom"
        value="<?php echo $_SESSION['profil']['nom']?> "/>
 <br/><label>prénom</label><br/>
 <input type="text" name="prenom"
        value="<?php echo $_SESSION['profil']['prenom']?>" />
 <br/><label>adresse</label><br/>
 <input type="text" name="adresse"
        value="<?php echo $_SESSION['profil']['adresse']?>" />
 <br/><label>code postal</label><br/>
 <input type="text" name="cp"
        value="<?php echo $_SESSION['profil']['cp']?>" />
 <br/><label>ville</label><br/>
 <input type="text" name="ville"
        value="<?php echo $_SESSION['profil']['ville']?>" />
 <br/><input type="submit"  value="enregistrer ma commande" />

</form>

</div>

</body>
</html>
```

Listing 15-26 : Le script enregistre_commande.php

```
<?php

include("variables.inc.php");

session_start();

if (empty($_REQUEST['nom']) || empty($_REQUEST['prenom']) ||
    empty($_REQUEST['adresse']) || empty($_REQUEST['cp']) ||
    empty($_REQUEST['ville']))
    die("ERREUR : tous les champs doivent être remplis.");

$liendb = mysql_connect($bddserver, $bddlogin, $bddpassword);
mysql_select_db ($bdd);

$date = date("Y-m-d G:i:s");

$i = 0;
foreach ($_SESSION['monpanier'] as $i) {
  $sql = "SELECT * FROM $table_produit WHERE idproduit = '$id'";
  $resultat = mysql_query ($sql);
  $produit = mysql_fetch_array ($resultat);
  $montant += $produit['prix'];
  $listeproduits .= ',' .  $produit['reference'];
```

```
  $i++;
}
$listeproduits[0] = ' ';
$montant += 5;

$sql = "INSERT INTO $table_commande (produits, montant,
✂ nom, prenom, adresse, date) VALUES ('"
✂ .$_REQUEST['listeproduits']."', '".$_REQUEST['montant'
✂ ]."', '".$_REQUEST['nom']."', '".$_REQUEST['prenom']."',
'".$_REQUEST['adresse']."\n".$_REQUEST['cp']."\n"
✂ .$_REQUEST['ville']."', '$date')";

mysql_query ($sql);

mysql_close($liendb);

session_unregister('monpanier');

$_SESSION['profil']['nom']     = $_REQUEST['nom'];
$_SESSION['profil']['prenom']  = $_REQUEST['prenom'];
$_SESSION['profil']['adresse'] = $_REQUEST['adresse'];
$_SESSION['profil']['cp']      = $_REQUEST['cp'];
$_SESSION['profil']['ville']   = $_REQUEST['ville'];

header("Location: $url/boutique.php");

?>
```

Dans *voir_caddie.php*, vous pouvez parcourir le contenu du panier directement avec la variable `$_SESSION['monpanier']`. Il en est de même pour le préaffichage des informations client. Tout est stocké dans la variable de session `$_SESSION['monprofil']`.

Vous remarquez l'usage d'une nouvelle fonction dans *enregistre_commande.php* : `session_unregister()`. Cette fonction permet tout simplement de supprimer la variable de session dont le nom est passé en paramètre.



La fonction `session_destroy()`

Cette fonction va plus loin que `session_unregister()` : elle vous permet de supprimer toutes les variables de session.

Les sessions sont donc un outil très puissant. Elles peuvent vous faciliter grandement la tâche pour le développement d'applications en ligne. Il ne faut cependant pas croire que les sessions pallient tous les défauts des

cookies. Une erreur courante consiste à se dire qu'avec les sessions vous n'avez plus à vous soucier des internautes qui n'acceptent pas les cookies. Ce raisonnement est complètement faux car les sessions utilisent en fait directement les cookies :

Listing 15-27 : La fonction session_id() retourne la valeur de l'identifiant de session unique qui a été alloué

```
<?php
session_start();
echo "Identifiant de session : ".session_id();
?>
```

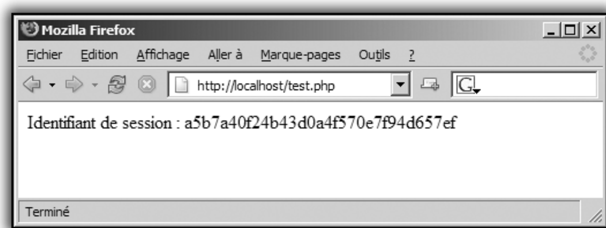


Figure 15.14 : *Valeur de l'identifiant de session*

Les variables de session sont en réalité stockées sur le serveur web et sont reconnues grâce à cet identifiant unique. Cet identifiant est stocké dans un cookie sur votre disque. Une personne qui n'accepte pas les cookies ne peut propager l'`id` de session qui lui est transmis, et ne peut donc pas profiter des sessions.

Ce mode de fonctionnement permet en revanche d'écarter les limitations des cookies concernant leur nombre et leur taille. Comme les données des sessions sont stockées sur le serveur, ces limitations sautent :

Listing 15-28 : Pour les plus curieux, la fonction session_save_path() retourne l'endroit où sont stockées les données des sessions sur le server (souvent /tmp sous Unix/Linux)

```
<?php
session_start();
echo session_save_path();
?>
```

Par défaut, une session est détruite quand l'internaute ferme son navigateur. Vous devinez donc que le cookie qui est créé pour stocker l'`id` de session est un cookie de session. Vous pouvez en avoir la preuve en utilisant la fonction `session_get_cookie_params()`, qui retourne un tableau contenant tous les paramètres du cookie de session :

Listing 15-29 : Affichage des propriétés du cookie de session

```
<?php
session_start();
print_r(session_get_cookie_params());
?>
```

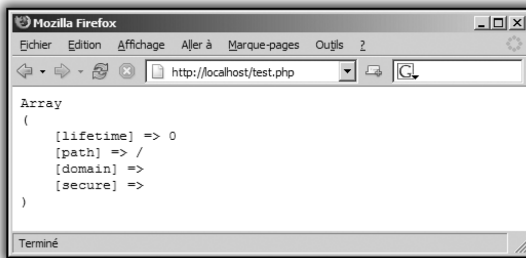


Figure 15.15 : *Propriétés du cookie de session*

Il est possible de faire en sorte qu'une session subsiste à la fermeture du navigateur. Utilisez pour cela la fonction `session_set_cookie_params()` qui permet de modifier les propriétés du cookie de session. Transformez ce petit exemple de compteur de manière à ce que la session soit conservée une semaine :

Listing 15-30 : Le compteur conserve désormais la bonne valeur, même si vous quittez le navigateur

```
<?php

session_set_cookie_params(time()+604800);
include("variables.inc.php");

session_start();

if (!isset($_SESSION['monpanier'])) $_SESSION['monpanier']
✂ = array();

$_SESSION['monpanier'][] = $_REQUEST['id'];

header("Location: $url/boutique.php?id=".$_REQUEST['id']);

?>
```

La fonction `session_set_cookie_params()` devant être appelée avant chaque appel à `session_start()`, l'idéal est de placer l'appel à ces deux fonctions dans *variables.inc.php*.

Listing 15-31 : Les sessions sont maintenant démarrées dans variables.inc.php

```
<?php

$bddserver  =  "localhost";
$bddlogin  =  "root";
$bddpassword  =  "";
$bdd  =  "test";
$table_eleve  =  "eleve";
$table_exam  =  "exam";
$url  =  "http://localhost";

session_set_cookie_params(time()+604800);
include("variables.inc.php");

?>
```

Cette option peut être intéressante si vous souhaitez mettre en place un système de panier permanent dans votre boutique : le client retrouve son panier lorsqu'il se reconnecte au site marchand.



Transmission de l'identifiant de session par URL

L'identifiant de session peut également être transmis par l'URL. Cette technique a l'avantage de fonctionner avec tous les internautes (qu'ils acceptent les cookies ou pas). Elle souffre cependant des défauts suivants :

- Elle est beaucoup plus lourde à mettre en place.
- Elle demande plus de ressources au niveau serveur.
- Elle nécessite que l'interpréteur PHP soit compilé avec certaines options.

15.3. Check-list

- Les cookies et les sessions permettent de propager des données liées à un internaute durant toute la durée de sa visite.
- Les sessions ont l'avantage d'être plus simples à gérer que les cookies. Une session est composée d'un cookie contenant son identifiant et d'un fichier sur le serveur contenant les données.
- Les variables `$_SESSION` et `$_COOKIE` sont utilisées pour y avoir accès.
- Longtemps décriés, les cookies sont désormais entrés dans les mœurs du Web.

Chapitre 16

La gestion de la sécurité

La sécurité avec PHP 485
Sécuriser les bases de données 493
Sécuriser le serveur web 496
Les outils d'analyse 503
Check-list 504

La popularité des applications web a non seulement attiré les utilisateurs et les développeurs, mais également les pirates informatiques (souvent appelés *hackers*). Les sites spécialisés en sécurité tels que **phpsec.org**, **secunia.com** ou **www.securityfocus.com** signalent désormais quotidiennement des failles dans des outils tels que le forum phpBB (**www.phpbb.com**), l'outil de gestion de contenu (CMS) PHP-Nuke (**phpnuke.org**), la plateforme d'e-commerce osCommerce (**www.oscommerce.com**), le gestionnaire de blog Wordpress (**wordpress.org**) et plusieurs centaines d'autres.

PHP ne souffre en lui-même d'aucun problème de sécurité. Le souci vient plutôt du fait qu'il est extrêmement facile de laisser des failles de sécurité au cœur d'applications écrites d'une part pour le Web et d'autre part en PHP. Listons quelques-uns des aspects les plus problématiques :

- PHP est un langage extrêmement facile à prendre en main. Un débutant sans connaissance préliminaire en sécurité peut réaliser et mettre en ligne un applicatif en quelques jours (voir en quelques heures).
- La sécurité d'une application web impose des connaissances informatiques relativement larges : serveur web, PHP, Javascript, authentification, base de données, sessions, cookies, protocoles, système d'exploitation (droits, chemins), etc.
- Pour une application en ligne, le nombre potentiel d'utilisateurs malintentionnés est aussi gigantesque qu'incontrôlable.
- Les techniques mises en œuvre pour exploiter les failles sont souvent triviales et, par conséquent, accessibles à de jeunes gens en mal d'amusement (les *scripts kiddies*). Nous sommes loin de la décompilation et de l'assembleur qui ne permettaient qu'à un tout petit nombre de pirates de « cracker » des logiciels.
- Certains défauts du langage PHP ont des implications directes dans la nature faillible des applications dont il est à l'origine : les nombreuses façons de transmettre des données (`input`, session, cookies), l'absence de « composants », l'inconsistance des différents noms (variables, objets, fonctions, etc.), la présence de variables globales rendant la relecture du code ardue et les tests d'unité pour le moins complexes.

Ce chapitre vise à vous présenter les failles potentielles les plus répandues qui peuvent s'immiscer au sein d'une application. Plus que

cela, il doit vous faire prendre conscience que la sécurisation d'un code est une étape indispensable et essentielle lors du développement d'un applicatif.

16.1. La sécurité avec PHP

Commençons par étudier les différents points devant être surveillés au sein de vos scripts PHP.

Le b-a ba

Certains développeurs ont tendance à concevoir leurs applicatifs en pensant qu'un certain nombre de tâches pourront être réalisées à la fin du projet. Cette pratique est généralement à proscrire car, comme dans beaucoup d'autres domaines, la fin d'un projet se déroule souvent dans l'urgence et le stress.

Il est donc conseillé de ne pas attendre l'issue du projet pour initialiser de manière cohérente vos différents mots de passe et ne pas laisser des accès de type `"admin/admin"`, `"test/test"`. Pour rappel, un mot de passe doit contenir au minimum huit caractères et inclure des caractères numériques.



Dictionnaires et brute force

N'allez surtout pas croire qu'un hacker souhaitant trouver le mot de passe de votre site essaiera successivement et péniblement plusieurs dizaines de possibilités. Il existe aujourd'hui des dictionnaires de mots et des scripts permettant en quelques minutes de tester plusieurs centaines de milliers de possibilités. Ces dictionnaires disposent également de tous les prénoms, marques d'alcools, noms de planètes et divers termes souvent utilisés en informatique pour initialiser un mot de passe.

L'échec d'une attaque lancée à l'aide d'un dictionnaire ne rebutera cependant pas un pirate motivé. L'attaque dite *brute force* consiste à tester toutes les combinaisons de caractères possibles les unes après les autres. Cette attaque peut durer des heures, voire des journées, mais n'oubliez pas que le pirate n'a pas à attendre devant sa machine, qu'aujourd'hui les machines sont extrêmement puissantes et que les liaisons Internet sont quasi gratuites.

Il peut donc être malin d'interdire un trop grand nombre de tentatives d'accès successives à une zone sécurisée. Il convient pour cela d'enregistrer chaque tentative (accompagnée de son adresse IP) dans une base de données et de vérifier, avant d'autoriser l'accès, qu'il n'y a pas eu plus de n tentatives depuis cette même adresse auparavant.

Mise à jour de PHP

Disposer d'un interpréteur PHP en permanence à jour est crucial pour votre applicatif. Chaque nouvelle version de PHP apporte en effet son lot de nouvelles fonctionnalités, de corrections de bugs et surtout de corrections de failles de sécurité. Bien que ces failles soient la plupart du temps mineures, il peut arriver que certaines soient exploitables de façon distante. Vous ne pourrez alors rien y faire, quelles que soient vos compétences de programmeur.

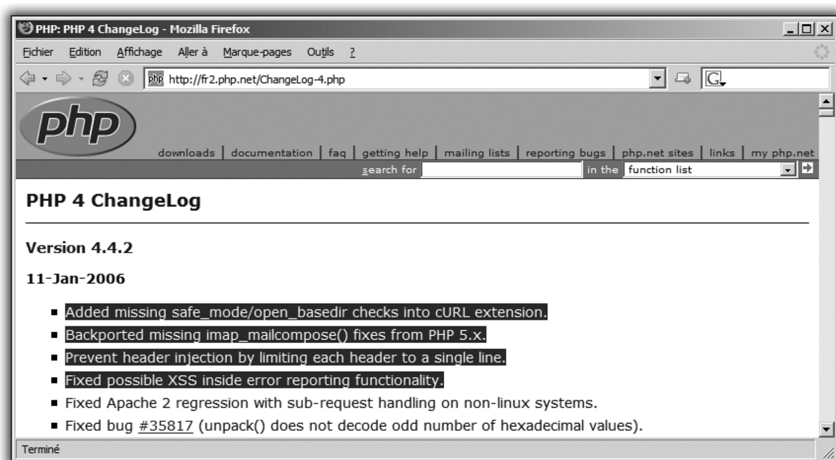


Figure 16.1 : *Exemple de lignes à prendre en compte dans le rapport de mises à jour d'une nouvelle version de PHP*

La fonction `phpversion()` ou la constante `PHP_VERSION` peuvent être utilisées pour obtenir la version de l'interpréteur.

Initialiser toutes les variables

Cette pratique consiste à donner une valeur à toutes les variables que vous utilisez au sein de votre script.

```
if ($_POST['identifiant']=='sys' && $_POST['pass']=='sys') {
  $status_admin = true;
}
```

Ce code n'est pas correct dans la mesure où la variable `$status_admin` n'a pas été initialisée.

La version correcte est la suivante :

```
$status_admin = false;
if ($_POST['identifiant']=='sys' && $_POST['pass']=='sys') {
  $status_admin = true;
}
```

Le statut d'administrateur est faux si l'authentification n'a pas été réalisée ou si elle échoue. Dans le premier code, son statut ne serait pas défini.



CURL

Les programmeurs débutants ont souvent tendance à penser que les tentatives de piratage ne passent que par la méthode `GET`. Cette supposition est complètement fausse. Des outils tels que CURL (**curl.haxx.se**) permettent ainsi de réaliser des scripts qui simulent l'envoi de paramètres en mode `POST`. CURL permet même de simuler la transmission de cookies. La page de documentation est disponible à l'adresse **http://curl.haxx.se/docs/manpage.html**.

De la même manière, le changement des mots de passe par défaut est essentiel. Ainsi n'oubliez surtout pas de modifier le mot de passe `root` de MySQL.

Utiliser les constantes

La plupart des attaques visent à exploiter une faille qui permettra de modifier une variable utilisée dans un endroit sensible du script (ex: inclusion, exécution etc.). L'utilisation des constantes, dont le contenu n'est pas modifiable par définition, permet de rendre la tâche du pirate plus ardue.

Se méfier de la puissance de certaines fonctions

La fonction `extract()` permet d'importer dans la table générale des variables le tableau qui lui est passé en paramètre. Voyons tout de suite un exemple qui devrait clarifier les choses :

Listing 16-1 : utilisation de la fonction extract()

```
print("bonjour : <br/>");
print("hello : <hr/>");
$tab = array('bonjour'=>'monde', 'hello'=>'world');
print_r($tab);
print("<hr/>");
extract($tab);
print("bonjour : $bonjour<br/>");
print("hello : $hello<br/>");
```

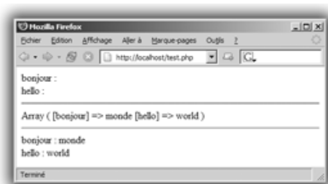


Figure 16.2 : Les cellules du tableau deviennent accessibles en tant que variables

Vous comprenez alors l'importance d'être sûr de son tableau avant d'appeler la fonction `extract()`. Un internaute qui parviendrait à modifier le tableau passé en paramètre pourrait écraser toutes les variables de l'application (par exemple `$status_admin`).

La fonction `extract()` peut également prendre un second paramètre qui indique le comportement à avoir quand une variable est déjà définie avant l'appel à `extract()`. Nous vous conseillons d'utiliser la valeur `EXTR_SKIP` qui interdit l'écrasement d'une variable préexistante.

```
$bonjour = 'coucou';
print("bonjour : $bonjour<br/>");
print("hello : $hello<hr/>");
$tab = array('bonjour'=>'monde', 'hello'=>'world');
print_r($tab);
print("<hr/>");
extract($tab,EXTR_SKIP);
print("bonjour : $bonjour<br/>");
print("hello : $hello<br/>");
```

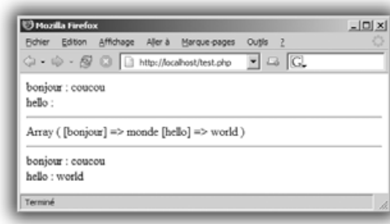


Figure 16.3 : *La variable $bonjour conserve sa valeur*

Dangers de la fonction mail

Il est courant de composer un mail à partir d'éléments provenant d'un formulaire. L'adresse email est souvent utilisée pour composer l'en-tête `From:` et permettre au destinataire du message, de répondre directement à l'internaute.

Listing 16-2 : Les paramètres du formulaire sont directement utilisés dans le mail

```
mail("fx@domain.fr","sujet","contenu","From:
✂ {$_REQUEST['email']}");
```

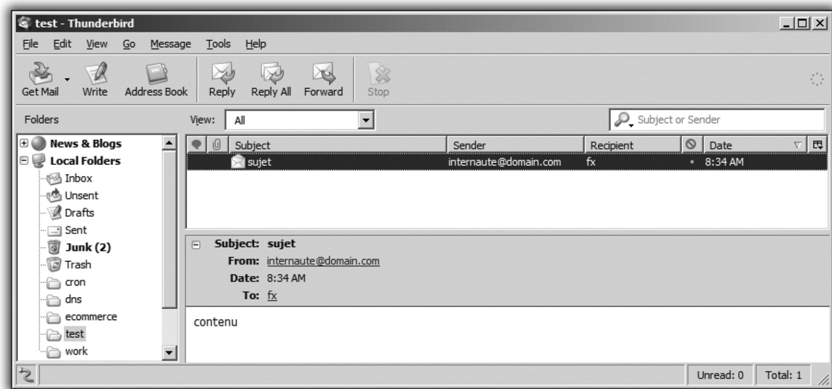


Figure 16.4 : *L'origine du destinataire apparaît bien : internaute@domain.com*

Notre erreur consiste ici à ne pas vérifier le paramètre avant de l'inclure dans le message. En effet, les spammers parviennent désormais à exploiter les millions de formulaires présents sur le web pour envoyer des messages non sollicités. Leur technique consiste à transmettre au script PHP non pas une adresse email mais une adresse email avec d'autres directives exploitables dans l'en-tête du mail.

En transmettant à notre script les paramètres suivants :

```
email=dest%40domain.com%0D%0ACc%3Adest2%40domain.com
```

le spammer initialise à la fois les en-têtes `From:` et `Cc:` du mail.

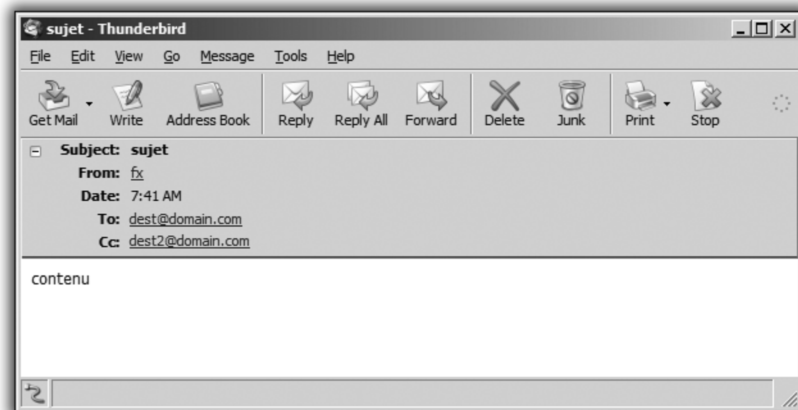


Figure 16.5 : *dest2@domain.com va recevoir un email de spam*

Pour être valide, notre script doit donc vérifier que le paramètre email ne contient pas de retour à la ligne.

Listing 16-3 : Vérification de la validité du paramètre email

```
if (strpos($_REQUEST['email'],"\n")!=false ||
    strpos($_REQUEST['email'],"\r")!=false)
  exit (1);
else
  mail("fx@domain.fr","sujet","contenu","From:
  ✂ {$_REQUEST['email']}");
```

Les cookies et les sessions

Aussi bien les cookies que les sessions prennent une place tous les jours plus importante dans les applications web en tant que point central du système d'authentification.

Il n'est donc pas étonnant que le cookie d'un internaute soit si convoité par les pirates. Il s'avère en plus que les techniques existent et qu'elles ne sont pas si complexes à mettre en place.

Le fonctionnement avec un passage de l'identifiant de session en paramètre est par exemple très dangereux. Prenez l'exemple d'un webmail fonctionnant sur ce schéma. Une URL typique au sein de

l'application pourrait être : www.monmail.com/lire.php?PHPSESSID=1234-5678&num_msg=ABC. Elle pourrait correspondre à la lecture du message ABC par le visiteur dont la session porte l'identifiant 1234-5678. Supposons maintenant que le courriel lu comporte dans son contenu un lien vers le site monsite.com et que l'internaute clique dessus. Si les développeurs du webmail ne sont pas vigilants, l'administrateur de monsite.com peut se retrouver dans ses logs avec une visite dont l'origine est précisément www.monmail.com/lire.php?PHPSESSID=1234-5678&num_msg=ABC. En cliquant sur ce lien qui dispose donc de l'identifiant de session, ce cher administrateur aura la surprise de pouvoir lire un courriel qui ne lui est pas du tout adressé. Pour parer à ce genre de faille, nous recommandons donc de passer par un script dont le rôle se limiterait à une redirection et qui ne prendrait en paramètre que l'URL de destination et en aucun cas l'identifiant de session ; par exemple : www.monmail.com/redir.php?url=http://www.monsite.com.

Ces techniques sont souvent évoquées en tant que *session hijacking*.



L'IP dans la session : une fausse bonne idée

Pour parer aux failles décrites précédemment, le programmeur peut avoir envie de mettre en œuvre un système permettant de vérifier si l'internaute qui réclame la session est le même que celui qui en est à l'origine. Une idée pourrait donc consister à placer dans la session l'adresse IP de l'internaute qui vient de la créer et à vérifier systématiquement si cette même IP est identique à celle de l'internaute qui réclame la session. Cette idée qui peut paraître bonne à première vue est hélas invalide. Il convient en effet de savoir qu'un visiteur ne garde pas obligatoirement la même IP durant toute la durée de sa visite. Certains routeurs permettent en effet la connexion à deux fournisseurs d'accès et autorisent ainsi un passage alterné par l'un ou l'autre des fournisseurs. Si le routeur est connecté à Free ADSL et à Noos, l'internaute, d'une page à l'autre, apparaîtra tour à tour avec une IP Free et une IP Noos. Même si cet aspect n'existe que pour 1 % des internautes, il est à prendre en compte.

Les transferts de fichiers

Le fait de demander une photo n'implique pas nécessairement que l'internaute choisira un fichier de type image. S'il envoie au contraire un fichier PHP, il pourra alors exécuter son script de manière distante au sein même de votre compte et provoquer de véritables catastrophes.

Le test suivant permet de vérifier que le fichier transmis se termine bien par *.jpeg*, *.jpg*, *.gif* ou *.png* :

Listing 16-4 : le fichier est-il de type image ?

```
if (preg_match("/\.(jpeg|jpg|gif|png)$/i",
               $_FILES['userfile']['tmp_name'])) {
    echo "ok";
}
else {
    echo "ko";
}
```

Inclusion de fichier

Soyez extrêmement prudent avec les inclusions de fichiers faisant intervenir des données passées en paramètres. L'exemple suivant est à proscrire :

```
if ($_REQUEST['action']!='accueil')
✂ include($_REQUEST['action']);
else include("index.php");
```

Un pirate peut transmettre la valeur suivante `action=../../WINDOWS/win.ini` et obtenir le contenu de n'importe quel fichier présent sur le disque dur.

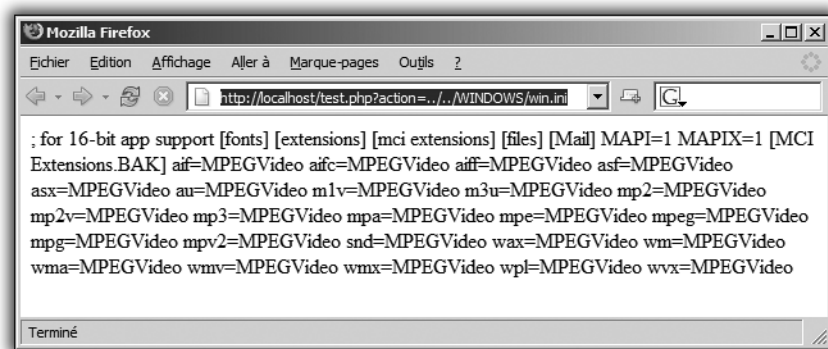


Figure 16.6 : *contenu du fichier win.ini présent dans C:\WINDOWS*

L'inclusion d'un fichier avec la fonction `include()` ne doit donc jamais exploiter directement une donnée transmise en paramètre.

16.2. Sécuriser les bases de données

Les failles liées aux bases de données sont aujourd'hui les plus répandues.

Les injections SQL

Le programmeur débutant ne réalise pas à quel point l'utilisation des bases de données impose une extrême prudence.

Étudiez le code suivant :

```
mysql_query("DELETE FROM matable WHERE ID = ".$_REQUEST
✂ ['monid']);
```

Cette ligne vise à supprimer de la table *matable* la ligne dont la clé est égale au paramètre `monid`. Si la variable `$_GET['monid']` contient la valeur `22`, la requête équivaut à ceci :

```
mysql_query("DELETE FROM matable WHERE ID = 22");
```

Supposons maintenant qu'une personne mal intentionnée transmette l'URL suivante : **http://127.0.0.1/supprime.php?monid=22+OR+ID+ %3E+0**.

Vous pouvez alors vous inquiéter car la requête devient :

```
mysql_query("DELETE FROM matable WHERE ID = 22 OR ID > 0");
```

Elle entraîne une suppression complète de la table *matable* ! Cette attaque est qualifiée d'injection SQL.

Deux habitudes importantes doivent être prises pour éviter cette situation :

- passer par la fonction `mysql_real_escape_string()` avant d'insérer des données extérieures dans la base (provenant d'un formulaire, d'un cookie etc.) ;
- utiliser les guillemets autour des données.

Listing 16-5 : version sécurisée de notre requête

```
mysql_query("DELETE FROM matable WHERE ID = '".
                    mysql_real_escape_string($_REQUEST
                    ['monid'])."'");
```

Les Cross Site Scripting

Ces attaques sont plus connues sous le terme générique de XSS. Le principe d'une attaque XSS est le suivant :

- Le pirate remplit un formulaire en y incluant des balises nocives. Les données du formulaire sont stockées dans une base de données sans être nettoyées préalablement.
- Un internaute accède au site (par exemple un forum), visualise l'article transmis par le pirate et subit l'assaut du pirate.

Les balises nocives dont nous parlons correspondent le plus souvent à des liens. Ces derniers sont écrits de manière à récupérer le cookie de l'internaute et à le transmettre au pirate.

Illustrons cela à l'aide d'un exemple. Le premier script enregistre les données dans une table *article* :

Listing 16-6 : ajout.php

```
<?php

if (!empty($_REQUEST['contenu'])) {
  $contenu = $_REQUEST['contenu'];
  $liendb = mysql_connect("localhost", "root", "");
  mysql_select_db("test");
  mysql_query("INSERT INTO article (contenu) VALUES
  ✂ ('$contenu')");
  print("enregistrement ok");
  return (true);
}

?>

<form method='post' action='ajout.php'>
<textarea style='width:100%' name='contenu'>
</textarea><br/>
<input type='submit' value='enregistrer'>
</form>
```

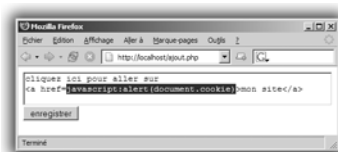


Figure 16.7 :
Texte renseigné par le pirate

Le second script permet d’afficher un article enregistré dans la base et de placer un cookie secret sur la machine de l’internaute.

Listing 16-7 : visualisation de l’article

```
<?php

setcookie('secret',rand(100,999));

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db("test");
$sql = "SELECT * FROM article";
$resultat = mysql_query ($sql);
$article = mysql_fetch_array ($resultat);

?>

<div style='width:200px; border:2px solid black;
padding:6px; height:200px'>

<?php print($article['contenu']); ?>

</div>
```

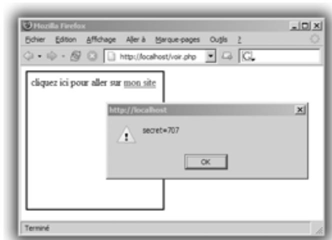


Figure 16.8 :
Apparition du cookie

En cliquant sur le lien de l’article, le code secret apparaît ! C’est grave car il va sans dire que dans la réalité les pirates font plutôt en sorte de se faire envoyer discrètement le code secret.

La parade classique pour ces attaques XSS consiste à protéger les données envoyées par les internautes à l’aide de la fonction `htmspecialchars()`. Cette fonction convertit les caractères `&`, `"`, `’`, < et > en leur équivalent HTML : `&`, `’`, `'`, `<` et `>`. Cela vous permet de ne pas altérer le contenu tout en vous protégeant.

Le script *ajout.php* doit donc être modifié de la façon suivante :

```
if (!empty($_REQUEST['contenu'])) {
  $contenu = htmlspecialchars($_REQUEST['contenu']);
  $liendb = mysql_connect("localhost", "root", "");
```

```
  mysql_select_db("test");
  mysql_query("INSERT INTO article (contenu) VALUES
  ✂ ('$contenu')");
  print("enregistrement ok");
  return (true);
}
```

Le même message est désormais enregistré de la manière suivante :

```
cliquez ici pour aller sur <a href=javascript:alert
✂ (document.cookie)>mon site</a>
```

Visuellement, vous obtenez un affichage correct et vous ne subissez plus cette attaque.

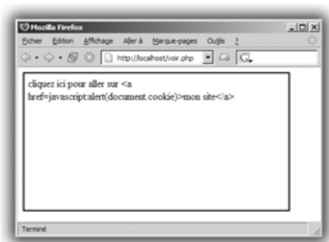


Figure 16.9 : *L'attaque apparaît au grand jour*

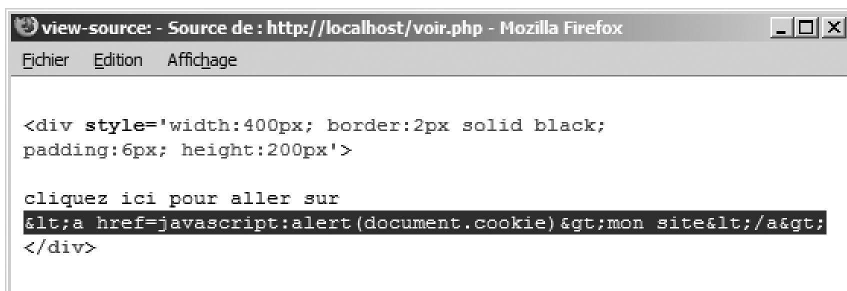


Figure 16.10 : *Les caractères dangereux ont bien été remplacés par leur équivalent HTML*

16.3. Sécuriser le serveur web

Le serveur web peut être assimilé au socle de votre application web. Un serveur web mal configuré, et c'est l'ensemble de votre outil qui peut en pâtir. Inversement, certains paramètres peuvent accroître largement la sécurité globale.

Les directives PHP

La plupart des directives de configuration de PHP peuvent être initialisées au sein du fichier *php.ini* ainsi qu'au niveau des fichiers de configuration d'Apache. Chaque site web du serveur peut ainsi disposer d'un environnement qui lui est propre.

Si certains paramètres par défaut ne vous conviennent pas, il est possible de demander un changement de configuration à votre administrateur système.

Un certain nombre de paramètres doivent être particulièrement contrôlés :

- `magic_quotes_gpc` : ce paramètre permet d'ajouter les fameux caractères d'échappement (`\'`, `\"`) pour les données transmises via les méthodes `GET`, `POST` ou via les cookies. Bien que cela soit a priori très pratique, cette fonctionnalité est risquée dans la mesure où elle ne conduit pas à un systématisme dans la protection des données. Les directives `magic_quotes_runtime` et `magic_quotes_sybase` peuvent également être forcées à `false` pour éviter les mêmes écueils.
- `register_globals` : cette directive est à `false` par défaut depuis la version 4 de PHP. Il est conseillé de la laisser ainsi afin d'éviter au maximum les problèmes d'initialisation et d'écrasement de variables rencontrés plus haut.
- `display_errors` : cette directive peut rester à `on` pendant tout le développement car elle vous permet d'afficher des informations précieuses durant la phase de correction des bugs (souvent qualifiée de phase de débogage) . Une fois le site mis en production, il est cependant conseillé de la passer à `off` afin de ne pas transmettre d'informations stratégiques à un éventuel pirate. Une erreur sur une inclusion de script donnera des informations sur l'organisation des répertoires et des scripts (il saura alors où frapper).

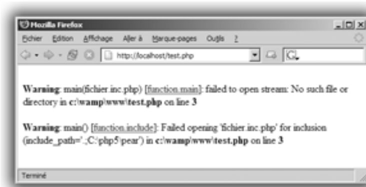


Figure 16.11 :
L'internaute sait désormais que test.php se situe dans C:\wamp\www

Une erreur sur une connexion mysql permettra quant à elle d'obtenir des informations sur le serveur SGBD (caché jusqu'à présent) et d'orienter certaines attaques sur celui-ci. Encore une fois, plus votre ennemi dispose d'informations à exploiter, plus sa capacité de nuisance sera grande et rapide.

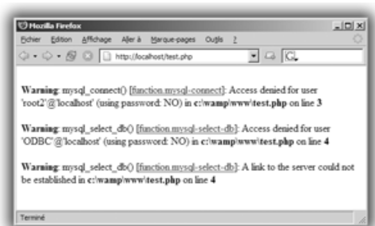


Figure 16.12 :
Erreur de connexion au SGBD



De la discrétion

Dans la même lignée, veillez à ce que le fichier de log créé par le serveur web ne soit pas accessible en ligne (par exemple **www.toto.com/logs/access.log**). Un tel fichier contiendrait en effet les mêmes informations que celles affichées en mode `display_errors on`.

La directive `display_errors` est directement liée à `error_reporting` qui définit le niveau de précision des affichages d'erreur. Il est ainsi possible d'indiquer à PHP de n'afficher que les erreurs ou de les afficher accompagnées d'avertissements (notice).

Prenez le script suivant :

```
print($tmp);
$tmp = 3;
print("ici");
```

Placez-vous tout d'abord dans un environnement où `error_reporting` n'affiche pas les avertissements :

```
error_reporting = E_ALL & ~E_NOTICE
```

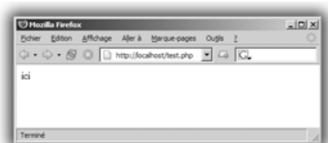


Figure 16.13 :
Rien à signaler a priori

Affichez maintenant les avertissements en plus des erreurs :

```
error_reporting = E_ ALL & E_NOTICE
```

Vous obtenez le résultat suivant :

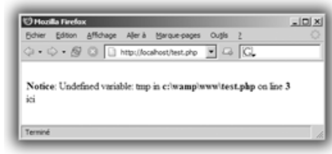


Figure 16.14 :
Vous êtes prévenu que la variable $tmp n'a pas été initialisée

En augmentant le degré d'affichage des erreurs, vous augmentez instantanément le taux de « capture » d'erreurs et la qualité globale du code.



La directive error_reporting

Cette directive peut prendre une multitude d'autres valeurs dont voici la liste :

- `E_ALL`, `E_ERROR` ;
- `E_WARNING` ;
- `E_PARSE`, `E_NOTICE` ;
- `E_CORE_ERROR` ;
- `E_CORE_WARNING` ;
- `E_COMPILE_ERROR` ;
- `E_COMPILE_WARNING` ;
- `E_USER_ERROR` ;
- `E_USER_WARNING` ;
- `E_USER_NOTICE`.

En jouant sur ces valeurs, vous obtiendrez un niveau de report en adéquation avec votre style de programmation.

Dans la même lignée, il est conseillé de logger tous les événements inhabituels qui ont lieu au sein de votre applicatif. La fonction `error_log()` vous permet d'ajouter vos propres commentaires au sein du fichier de log de PHP.

```
error_log_test("ajout de mes propres logs");
```

Vous retrouvez ces commentaires dans le fichier *C:\wamp\logs \php_error.log*.

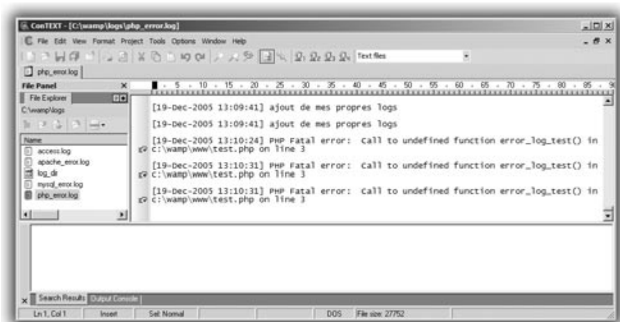


Figure 16.15 : *Vos logs apparaissent au milieu d'autres lignes d'erreur*

Cette fonction est particulièrement adaptée pour signaler des comportements anormaux. Voyez des exemples d'événements pouvant être loggés :

- un échec répété lors de l'authentification ;
- un internaute qui souhaite régler un panier qui n'existe pas ;
- une erreur dans une requête SQL.



Fonction error_log() et débogage

L'avantage de l'utilisation de la fonction `error_log()` par rapport à une simple fonction `print()` pendant une phase de débogage est de ne pas entrer en conflit avec les fonctions `setcookie()`, `header()` ou `session_start()`. En effet, l'utilisation de `print()` avant ces fonctions entraînerait l'affichage d'un message d'erreur. La fonction `error_log()` qui n'affiche rien à l'écran, mais qui écrit dans le fichier de log, ne rencontre pas ce problème.

En plus du message, il peut être intéressant d'ajouter l'adresse IP de l'internaute qui est à l'origine de l'événement.

```
error_log_test($_SERVER['REMOTE_ADDR']" : bizarre !");
```

S'il s'agit d'un applicatif où l'internaute est identifié, l'ajout de l'identifiant à la ligne de log est également pertinent.

Les directives Apache

La présence d'un répertoire contenant tous les fichiers inclus (`include()`) est une situation extrêmement courante. Nous avons

évoqué, dans un chapitre précédent, un moyen d'éviter l'appel direct de ces fichiers en les protégeant avec *indentification.inc.php*. Une solution moins portable, mais largement plus sûre, consiste à faire en sorte qu'Apache refuse l'accès direct à ce répertoire. Si vous souhaitez protéger le répertoire *includes* situé dans *C:/wamp/www*, la directive est la suivante :

```
<Directory "C:/wamp/www/includes">
   order deny,allow
   deny  from all
</Directory>
```



Figure 16.16 : *Message renvoyé par Apache*

L'accès direct aux fichiers inclus est particulièrement dangereux lorsque le développeur utilise une extension non reconnue par Apache. Imaginez que le répertoire *includes* contienne le fichier de configuration général de l'application : *config.inc*. Si Apache n'a pas été paramétré pour considérer les extensions **.inc* comme des fichiers PHP, vous obtenez ce résultat catastrophique :

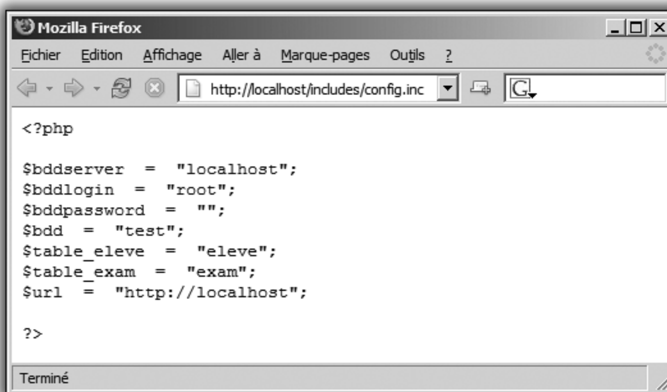


Figure 16.17 : *Le fichier include non protégé*



Accès en clair

La recherche suivante `"mysql filetype:inc"` sur Google vous montre à quel point il peut être simple de découvrir les identifiants et les mots de passe d'accès à une multitude de bases de données.

Il convient donc de vérifier auprès de votre administrateur système si l'extension choisie pour les includes peut convenir.



Ajout d'extensions dans Apache

La directive `AddHandler` permet de définir les extensions des scripts PHP : `AddHandler application/x-httpd-php .php .php3 .inc`.

Il conviendra également de vérifier qu'aucun répertoire sensible ne soit listé. Apache permet en effet de lister l'ensemble des fichiers contenus dans un répertoire si ce répertoire contient l'option `Indexes` :

```
<Directory "C:/wamp/www/testindexes">
 Options Indexes
</Directory>
```

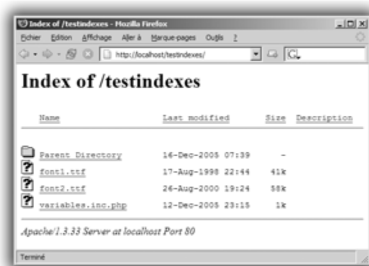


Figure 16.18 :
Exemple de liste de fichiers



Fichier index

La liste des fichiers n'est proposée que si le répertoire ne contient pas de fichier *index* (par exemple *index.html*, *index.htm*, *index.php*, etc.). La liste de ces fichiers peut être modifiée dans le fichier *httpd.conf* avec la directive `DirectoryIndex`.

L’ajout des directives suivantes en bas du fichier de configuration d’Apache (*httpd.conf*) permet d’interdire l’accès au répertoire *testindexes* :

```
<Directory "C:/wamp/www/testindexes">
 Options -Indexes
</Directory>
```



Figure 16.19 :
Le répertoire ne peut plus être "listé"

La sécurité HTTPS

HTTPS correspond au protocole HTTP complété d’une couche SSL (Secure Socket Layer) de cryptage des données. Cette protection est particulièrement utile pour lutter contre les attaques de *sniffing*. Le *sniffing* peut être rapproché d’une écoute des données transitant entre votre navigateur et le serveur web. Sans la couche SSL, ces mêmes données circulent en clair sur le réseau. Lorsqu’il s’agit d’une page d’informations, le risque est inexistant. Lorsqu’il s’agit en revanche d’une authentification, votre identifiant et votre mot de passe apparaissent en clair sur l’écran du pirate. En configurant l’extension SSL d’Apache (`mod_ssl`), vous vous assurez de l’anonymat de vos échanges.

Cette technique se révèle superflue dans 90 % des cas. Le point central du *sniffing* consiste à pouvoir se « brancher » sur la liaison Internet de la cible. Or, ce n’est possible que si l’attaquant se trouve dans un réseau local, non loin de la machine cible.

16.4. Les outils d’analyse

Compte tenu de l’importance que prennent les applications web dans l’économie moderne, certains éditeurs ont sauté sur l’occasion pour proposer des outils permettant de les auditer.

Le logiciel Acunetix Web Vulnerability Scanner (**www.acunetix.com/wvs**) par exemple, permet à partir d'une simple adresse web, de tester les attaques sur des failles du langage, les injections SQL, les attaques de type XSS (Cross Site Scripting), les attaques concernant les passages de paramètres, les formulaires d'authentification (en veillant à ce que les identifiants et les mots de passe ne soient pas trop simples à trouver), ainsi qu'une dizaine d'autres attaques classiques.

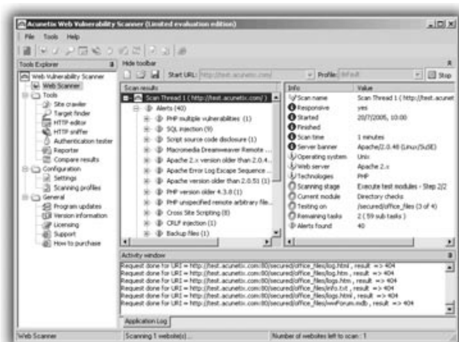


Figure 16.20 :
Exemple d'interface du logiciel Acunetix Web Vulnerability Scanner

16.5. Check-list

La sécurité est un domaine complexe. Plusieurs règles doivent systématiquement rester à votre esprit durant la phase de développement :

- Les paramètres reçus dans un script peuvent être forcés par un utilisateur malicieux qui composerait sa propre *Query String* ou qui modifierait ses cookies. Les paramètres reçus dans un script doivent donc toujours être vérifiés avant d'être exploités.
- La vérification doit être encore plus poussée lorsqu'il s'agit de composer une requête SQL à partir de paramètres.
- Il est important de logger les événements étranges et incohérents qui ont lieu au sein de votre applicatif. Cela sera votre première source d'information en cas d'attaque.
- Les rapports d'erreur doivent être le plus complets possible durant la phase de développement. Ils vous permettront notamment de détecter les variables non initialisées.

Chapitre 17

Les trucs et astuces

PHP 506
MySQL 523
HTML et Javascript 525

Comme pour tout langage de programmation, il est possible avec l'expérience d'améliorer et d'optimiser son code. Ce chapitre va vous permettre de présenter quelques fonctionnalités et syntaxes peu connues de PHP et de MySQL.

17.1. PHP

Définir autrement une chaîne de caractères

Quand vous souhaitez définir des chaînes de caractères multilignes, il est courant d'écrire ceci :

```
$machaine = "ligne1\n";
$machaine .= "ligne2\n";
$machaine .= "ligne3";
```

Une syntaxe alternative (très proche de celle du Perl) est proposée par PHP et peut se révéler plus lisible. Elle consiste à englober les lignes entre des délimiteurs, par exemple `<<<EOS` et `EOS;` :

```
$str = <<<EOS
ligne1
ligne2
ligne3
EOS;
```

Le motif `EOS` n'est pas obligatoire et peut être remplacé, par exemple par `<<<END`… `END;`. Il faut cependant veiller à conserver le même motif pour l'ouverture et la fermeture.

La balise de fermeture doit impérativement être isolée sur une ligne. Il n'est donc pas question d'écrire ceci :

```
...
ligne2
ligne3 EOS;
```

Le texte peut en revanche contenir des variables :

```
$x = 1;
$y = 2;

$str = <<<EOS
liste des variables :
- x = $x
- y = $y
EOS;
```

Pour inclure dans le texte la valeur d'un tableau, la syntaxe d'accès à l'élément du tableau est un peu particulière :

```
$tab = array("v1" => 3, "v2" => 4);
```

```
$str = <<<EOS
v1 = {$tab['v1']}
v2 = {$tab['v2']}
EOS;
```

Il n'est pas indispensable de passer par une variable intermédiaire pour utiliser cette syntaxe, l'instruction suivante est tout à fait valide :

```
echo <<<EOS
ligne1
ligne2
ligne3
EOS;
```

Cette syntaxe est extrêmement pratique quand vous devez afficher un long texte contenant de nombreuses variables.



HTML brut et `echo`

Des tests ont montré qu'il n'était pas vraiment plus rapide d'écrire du code HTML brut que de passer par un `echo`. Les deux syntaxes suivantes sont donc aussi rapides :

```
valeur de la variable
x = <?php echo $x; ?>
```

```
echo <<<EOS
valeur de la variable
x = $x
EOS;
```

Cette notation est souvent qualifiée de *here printing*.

Raccourcir un if... else...

Il est possible, pour des `if… else…` très courts, de remplacer la syntaxe standard par la suivante :

```
(instruction 1)?(instruction 2):(instruction 3);
```

Si la première expression retourne `true`, c'est la deuxième instruction qui est exécutée, sinon c'est la troisième expression.

```
$x = 4;
($x > 5)? print "x ($x) est supérieur à 5" : print "x ($x)
✂ est inférieur à 5";
```

Si vous aviez voulu utiliser la fonction `echo`, vous auriez pu écrire :

```
$x = 4;
echo ($x > 5)?"x ($x) est supérieur à 5":"x ($x) est
✂ inférieur à 5";
```

L'exemple suivant va permettre de préremplir une `CHECKBOX` en fonction de la valeur prise par une variable :

Listing 17-1 : présélection d'une case à cocher

```
<?php

$ln = "fr";

?>

<form>

français :
<input type='radio' <?php echo ($ln ==
✂ "fr")?"checked='true'":""; ?>
       name="fr" value="fr" /><br/>

étranger :
<input type='radio' <?php echo ($ln !=
✂ "fr")?"checked='true'":""; ?>
       name="fr" />

</form>
```

L'autre syntaxe des structures de contrôle

Il est courant d'avoir à écrire un code de ce type :

```
if ($reponse == "OK")
{
  echo "le message a bien été envoyé<br/>";
  echo "merci de votre visite";
}
else
{
  echo "problème lors de l'exécution<br/>";
  echo "merci de recommencer";
}
```

Les blocs de code entre les `if`… `else`… ne contiennent que du texte brut.

PHP propose une syntaxe alternative, plus élégante, pour parvenir au même résultat :

Listing 17-2 : Les blocs HTML et PHP sont plus clairement séparés

```
<?php if ($reponse == "OK") : ?>

le message a bien été envoyé<br/>
merci de votre visite

<?php else: ?>

problème lors de l'exécution<br/>
merci de recommencer

<?php endif; ?>
```

La nouvelle syntaxe est donc la suivante :

```
if () : ... else () : ... endif;
```

Cette syntaxe est disponible pour les autres structures de contrôle :

```
if() : ... elseif () : ... else () : ... endif;
while () : ... endwhile;
for () : ... endfor;
```

Raccourcir un simple bloc echo

Il est courant de devoir utiliser un bloc PHP uniquement pour afficher une variable :

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>">
```

PHP propose une syntaxe alternative pour y parvenir :

```
<?=$nom_variable?>
```

Cet exemple devient alors :

```
<form action="<?=$_SERVER['PHP_SELF']?>">
```

Pour afficher deux valeurs, vous pouvez écrire :

```
<?php

$x = 2;
$y = 3;

?>

Valeurs de x, y =  <?="$x, $y"?>
```

L'avantage de cette syntaxe est donc de réduire votre code et de le rendre plus lisible. Plus un fichier est court et plus vous pouvez

rapidement en avoir un aperçu d'ensemble. Cette syntaxe permet également de séparer plus clairement les zones de texte brut des zones de code.

Cette syntaxe dépend de la directive de configuration `short_open_tag` présente dans *php.ini*. Cette directive doit être passée à `on` pour permettre l'utilisation de ces *short tags*. Les utilisateurs de Wamp Server devront notamment faire la modification.

Donner une valeur par défaut à un paramètre d'une fonction

Nous avons vu que la syntaxe pour déclarer une fonction est la suivante :

```
function nom_fonction($param1,$parma2)
{
  bloc_de_code;
}
```

Il est possible de donner des valeurs par défaut aux paramètres de la fonction en la déclarant ainsi :

```
function nom_fonction($param1 = "valeur_par_défaut")
{
  bloc_de_code;
}
```

Écrivez une fonction `affiche_retour()` et appelez-la de deux façons différentes :

```
<?php

function affiche_retour($url = "/")
{
  echo "<a href=$url>retour</a><br/>";
}

// version 1 : nous utilisons le paramètre par défaut
echo "merci de votre visite<br/>";
affiche_retour();

//version 2 : nous passons la valeur du paramètre
echo "merci de votre visite<br/>";
affiche_retour("/");

?>
```

Attention, si la fonction possède plusieurs paramètres, seul le dernier pourra prendre une valeur par défaut !

```
<?php

function affiche_retour($url,$nomlien = "retour")
{
  echo "<a href=$url>retour</a><br/>";
}

//version 2 : valeur par défaut pour le deuxième paramètre
echo "merci de votre visite<br/>";
affiche_retour("/");

//version 2 : nous passons les valeurs des 2 paramètres
echo "merci de votre visite<br/>";
affiche_retour("/","back");

?>
```

Transmettre un nombre variable de paramètres à une fonction

PHP propose un moyen de passer un nombre variable de paramètres à une fonction. Deux fonctions doivent être utilisées pour y parvenir :

- `func_num_args()` : retourne le nombre d'arguments passés en paramètres.
- `func_get_arg()` : permet d'accéder par index à chacun des paramètres.

Listing 17-3 : affiche_param() peut recevoir un nombre indéfini de paramètres

```
<?php

function affiche_param()
{
  $nb_param = func_num_args();
  for ($i=0;$i<$nb_param;$i++) {
    print("paramètre $i : " . func_get_arg($i) . "<hr>");
  }
}

affiche_param("a",2,"e","i",6);

?>
```

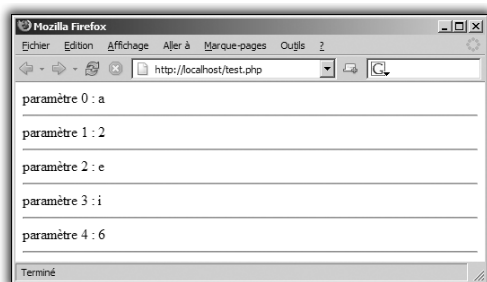


Figure 17.1 : *Fonctions avec un nombre indéfini de paramètres*

Utiliser un opérateur de comparaison de type

PHP rend la gestion des types des variables extrêmement simple. À trop simplifier les choses cependant, le risque est parfois de commettre des erreurs.

Étudiez le code suivant :

Listing 17-4 : comparaison hasardeuse

```
$x = 0;
$y = false;

if ($x == $y) echo "x et y sont égaux";
else echo "x et y sont différents";
```

Ce script va afficher `"x et y sont égaux"`, bien que `$x` soit un numérique entier et `$y` un booléen. Cela peut convenir la plupart du temps, car PHP place souvent au même niveau `1/true` et `0/false`.

Ce comportement peut cependant devenir gênant quand `0` et `false` doivent représenter deux valeurs différentes. La fonction suivante retourne, par exemple, la conversion d'un nombre de jours en secondes, et `false` en cas d'erreur :

```
function jour_sec($jour)
{
  if ($jour < 0)
  {
    return false;
  }
  else
  {
```

```
    return $jour * 24 * 60 * 60;
  }
}
```

Quand vous appelez la fonction avec la valeur `0`, la valeur de retour est `0`. Quand vous l'appelez avec la valeur `–12`, c'est la valeur `false` qui est cette fois retournée. Pour pouvoir faire la différence entre ces deux valeurs, PHP met à votre disposition certains opérateurs de comparaison sur les types :

Tableau 17.1 : *Opérateurs de comparaison sur les types*

Opérateur	Résultat
`$a === $b`	`$a` et `$b` sont égaux et de même type
`$a !== $b`	`$a` et `$b` sont différents ou de types différents

Vous pouvez donc faire appel à votre fonction de la manière suivante :

```
$nbsec = jour_sec(0);
if ($nbsec === false) echo "le paramètre n'est pas valide";
else echo "nombre de secondes : $nbsec";
```

Le script affiche ici `"nombre de secondes : 0"`.

Les attributs __FILE__ et __LINE__

Au cours de l'exécution d'un script PHP, `__FILE__` et `__LINE__` contiennent en permanence le chemin du fichier dans le lequel vous vous trouvez et le numéro de ligne en cours.

Listing 17-5 : Utilisation de __FILE__ et __LINE__

```
<?php
print("Bonjour vous vous trouvez dans le fichier <b>");
print(__FILE__);
print("</b> à la ligne <b>");
print(__LINE__);
print("</b>.");
?>
```



Figure 17.2 : *Résultat*

Les variables variables

Le nom d'une variable peut être lui-même une variable. Si $a contient 1 et $b contient 'a', la variable variable $$b correspond en fait à $a.

Listing 17-6 : $$b est une variable variable

```
<?php
$a = 1;
$b = 'a';
echo $$b; // affiche 1
?>
```

Les opérateurs sur les tableaux

Tableau 17.2 : *Opérateurs sur les tableaux*

Opérateur	Résultat
$a + $b	Union de $a et de $b
$a == $b	Renvoie true si $a et $b sont composés des mêmes paires clé/valeur ; les tableaux sont alors dits égaux
$a === $b	Comme ==, avec des vérifications en plus sur l'ordre et le type des données ; les tableaux sont alors dits identiques
$a != $b	Renvoie true si $a et $b ne sont pas égaux
$a <> $b	Renvoie true si $a et $b ne sont pas égaux
$a !== $b	Renvoie true si $a et $b ne sont pas identiques

L'opérateur d'union a cela de spécial que les clés qui seraient présentes dans $a et $b ne sont pas réécrites.

```
$a = array("couleur1"=>"rouge","couleur2"=>"vert");
$b = array("couleur1"=>"jaune","noir","vert");
print_r($a+$b);
```

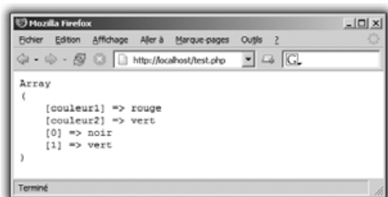


Figure 17.3 : *Union de $a et $b*

Les techniques d'optimisation en PHP

L'optimisation d'un code ne doit jamais être réalisée au détriment de sa clarté et de sa lisibilité. Cette règle est d'autant plus vraie si vous débutez en programmation ou si vous ne travaillez pas de manière isolée sur le projet.

Il est cependant utile de connaître quelques techniques d'optimisation, surtout si votre projet nécessite une rapidité extrême, et qu'il doit être utilisé par un grand nombre de personnes. Un espace mémoire de 100 ko gaspillé par un script peut en effet se transformer en plusieurs mégaoctets en cas d'afflux important de visiteurs.

Limiter le nombre de variables

Si vous avez besoin de nombreuses variables temporaires et que celles-ci ne soient utilisées que localement (pour une seule instruction, par exemple), essayez de conserver la même variable tout au long du script. Une telle variable est souvent appelée `$tmp`.

Listing 17-7 : Utilisation de deux variables temporaires : $heures et $minutes

```
$jours = 10;

$heures = 10 * $jours;
echo "nombre de d'heures : $heures";

$minutes = $heures * 60;
echo "nombre de minutes : $minutes";
```

Listing 17-8 : Utilisation d'une seule variable temporaire : $tmp

```
$jours = 10;

$tmp = 10 * $jours;
echo "nombre de d'heures : $tmp";

$tmp *= 60;
echo "nombre de minutes : $tmp";
```

Vous avez vu qu'une bonne habitude en programmation consiste à passer en variables un grand nombre de paramètres utilisés par vos scripts (notamment tous les paramètres susceptibles d'être modifiés lors d'un changement d'hébergeur). Il ne faut cependant pas tomber dans l'excès inverse et passer toutes les données (tous les paramètres) en

variables. Une variable est en effet stockée en mémoire, et plus vous en avez dans votre code, plus celui-ci consomme de mémoire lors de son exécution.

Utiliser les bons caractères

Quand une chaîne de caractères ne contient pas de variables, utilisez les simples primes :

```
$ch1 = 'bonjour';
$ch2 = "valeur de ch1 : $ch1";
```

En fonctionnant ainsi, vous évitez à l'interpréteur PHP d'essayer de remplacer les variables par leur valeur et vous économisez des ressources.

Limiter les requêtes aux bases de données

C'est de loin l'optimisation qui vous sera le plus utile. Supprimer une requête superflue est largement plus important que toutes les optimisations syntaxiques imaginables. Une requête est en effet très consommatrice en temps et en mémoire.

Si vos requêtes sont indispensables, essayez alors de les optimiser en utilisant des index sur vos tables ou en choisissant des critères de sélection très stricts. Spécifier les colonnes utiles, plutôt qu'un astérisque (`*`) dans un `SELECT`, est également une excellente habitude à prendre.

Les jointures intelligentes et les requêtes imbriquées proposées par MySQL 5 permettent très souvent de limiter le nombre requêtes.

Les fonctions include() et require()

La fonction `include()` n'est pas la seule à permettre de faire appel à des données contenues dans un fichier extérieur. La fonction `require()` peut aussi être utilisée dans ce cas.

La différence entre ces deux fonctions réside dans le fait que le code appelé par `require()` est inclus dans tous les cas et n'est exécuté qu'une fois.

De ce fait, seule la fonction `include()` peut être utilisée dans une boucle :

Listing 17-9 : Le script test.php

```
<?php

for ($i=0;$i<=10;$i++)
{
  $double = include("double.inc.php");
}

?>
```

Listing 17-10 : Le script double.inc.php

```
<?php

return $i * 2;

?>
```

L'exemple précédent présente un autre avantage de la fonction `include()` sur la fonction `require()` : la possibilité de récupérer dans le script appelant (*test.php*) une valeur retournée par le script appelé (*double.inc.php*). Cette valeur de retour correspond à la valeur retournée (`return()`) par le code inclus (retour qui a mis fin, par ailleurs, à l'exécution de ce même code).

Les fonctions `include()` et `require()` peuvent faire appel à un fichier distant. Il est ainsi tout à fait autorisé d'écrire ceci :

```
include("http://www.server.com/fonction.txt");
```

Le code PHP contenu dans le fichier *fonction.txt* sera alors inclus dans le script (le code doit être entouré des balises `<?php` et `?>`). Si en revanche vous faites appel à un fichier *.php*, ce n'est pas le code que vous allez inclure dans votre script, mais le résultat de l'interprétation du code par le serveur HTTP distant. Si ce code fait appel à des variables globales du serveur, le résultat sera alors différent.

Une autre fonction peut se révéler intéressante : `include_once()`. Cette fonction permet de n'inclure un fichier qu'une fois et évite ainsi de provoquer des erreurs si le fichier inclus se connecte à une base ou crée un objet (dans ces deux cas, l'interpréteur indiquerait que la ressource existe déjà).

L'affichage tampon : output buffering

L'*output buffering* est une technique qui permet de ne pas afficher directement les données à l'écran, mais plutôt de les stocker temporairement dans un tampon intermédiaire (*buffer*).

Trois fonctions sont utilisées pour l'*output buffering*.

- `ob_start()` : pour initialiser l'*output buffering*.
- `ob_get_contents()` : pour récupérer le contenu du *buffer* dans une variable.
- `ob_end_clean()` : pour arrêter l'*output buffering* et vider le *buffer*.

Considérez l'exemple suivant :

```
<?php
ob_start();
echo "toto";
$output = ob_get_contents();
ob_end_clean();
echo "titi";
echo $output;
?>
```

Ce script n'affiche pas `tototiti` mais `tititoto`. L'affichage de `toto` a été fait dans le *buffer*, et ce *buffer* n'est ensuite affiché qu'après `titi`.



Entrelacement d'output buffering
Un `ob_start()` peut être inclus au sein d'un autre `ob_start()`.

Si vous aviez utilisé la commande `readfile()` entre `ob_start()` et `ob_end_clean()`, le contenu du fichier aurait aussi été placé dans le *buffer*.

Grâce à cette technique, il devient possible de préparer dans un *buffer* un contenu complexe provenant de sources multiples (fichier, bases de données) et de n'afficher le *buffer* que si toutes les opérations se sont bien déroulées. En fonctionnant ainsi, vous évitez de vous retrouver avec des erreurs en plein milieu de la page.

L'*output buffering* associé à la lecture de fichier distant peut aussi être très intéressant pour la gestion de *templates*.

Fin de bloc PHP

Lorsque votre script se termine par un bloc PHP, l'ultime balise de fermeture `?>` devient facultative. Cette particularité a surtout l'avantage d'éviter de provoquer une erreur lorsque l'inclusion d'un fichier se terminant par `?>` précède un appel à une des fonctions suivantes : `session_start()`, `header()` ou `setcookie()`. Ces fonctions ne peuvent en aucun cas être précédées d'un affichage. Or, si vous laissez ne serait-ce qu'un retour à la ligne ou un espace derrière `?>`, une erreur sera déclenchée.

Le paramètre caché de break et continue

Le chapitre consacré à la syntaxe avait présenté les instructions `break` et `continue` qui permettent, pour la première, de stopper l'exécution d'une boucle et pour la seconde, de forcer le passage à l'itération suivante.

Dans le cadre de boucles imbriquées, ces deux instructions peuvent prendre en paramètre une valeur numérique. Pour le `break`, cette valeur correspond au nombre de boucles qui seront arrêtées et pour le `continue`, au nombre de boucles qui seront ignorées.

Listing 17-11 : Exemple illustrant l'utilisation d'un paramètre avec break et continue

```
$tab = array("a","b","c","d","e");

foreach ($tab as $lettre) {
  print("<br/> $lettre : ");
  for ($i=0;$i<=5;$i++) {
    if ($lettre=="a" && $i==1) continue 1;
    if ($lettre=="b" && $i==1) continue 2;
    if ($lettre=="c" && $i==2) break 1;
    if ($lettre=="d" && $i==2) break 2;
    print($i);
  }
}
```

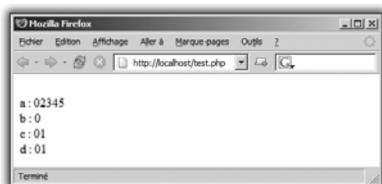


Figure 17.4 : *Break et continue*

Le fonctionnement de cet exemple et le suivant :

- si `$lettre == a` et `$i == 1` : nous continuons dans la boucle `for` en ignorant uniquement l'affichage de la valeur `1`,
- si `$lettre == b` et `$i == 1` : nous passons à l'itération suivante de la boucle de niveau supérieur : `foreach`,
- si `$lettre == c` et `$i == 2` : nous arrêtons la boucle en cours : `for`,
- si `$lettre == d` et `$i == 2` : nous arrêtons la boucle au niveau supérieur : `foreach` et ne passons pas au traitement de la lettre `"e"`.

Chaîne de caractères sous forme de tableau de caractères

Les chaînes de caractères peuvent être considérées comme des tableaux scalaires. Le premier caractère ayant comme d'habitude l'indice `0`.

Listing 17-12 : Affichage de chacun des caractères de la chaîne $str

```
$str = "Bonjour Mr Gueudet";
print("$str<hr/>");

$n = strlen($str);
for ($i=0;$i<$n;$i++) {
  print("str[$i] = ".$str[$i]."<br/>");
}
```



Figure 17.5 : *Résultat de l'affichage*

Rendre disponible un site wamp sur internet

Dans 99 % des cas, un site web fonctionnant avec Wamp ne sera pas accessible sur une autre machine. Pour rendre le site disponible, une modification doit être réalisée au niveau du fichier de configuration d'Apache (`httpd.conf`). La ligne `Allow from 127.0.0.1` doit être remplacée par `Allow from all`.

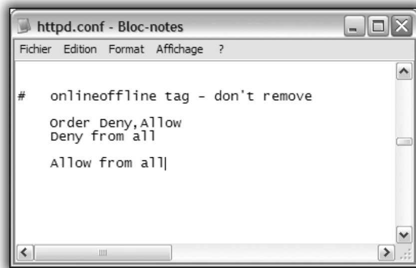


Figure 17.6 : *Modification de la configuration d'Apache*

Ce changement indique que toutes les machines (`All`) peuvent désormais accéder à Apache et plus uniquement la machine sur laquelle est installée Wamp (`127.0.0.1`).

Un redémarrage d'Apache doit être réalisé pour prendre en compte la modification.

Pour accéder au site web, une machine du réseau local devra utiliser l'adresse IP de la machine (ex. `http://192.168.0.2/test.php`) plutôt que localhost (`http://localhost/test.php`). Cette adresse IP peut être trouvée en tapant la commande `ipconfig` dans un terminal.

```
Carte Ethernet Connexion réseau sans fil:

        Suffixe DNS propre à la connexion :
        Adresse IP. . . . . . . . . . . . : 192.168.0.2
        Masque de sous-réseau . . . . . . : 255.255.255.0
        Passerelle par défaut . . . . . . : 192.168.0.253
```

Figure 17.7 : *L'adresse IP est ici 192.168.0.2*

L'étape suivante consiste à permettre à des ordinateurs situés en dehors du réseau local d'accéder au site. Il convient pour cela de modifier des paramètres au niveau de votre routeur. La Freebox permettra d'illustrer ces interventions avec un cas concret.

La première étape consiste à accéder à l'espace *Mon Compte* du site **www.free.fr**. Une fois identifié, la rubrique *Fonctionnalités optionnelles de la Freebox* permet de configurer le routeur (*Fonction routeur*). L'objectif est alors de créer une redirection du port `80` (`TCP`) du routeur vers le port `80` de votre machine. Là encore, l'IP de votre machine (ex. `192.168.0.2`) est nécessaire.

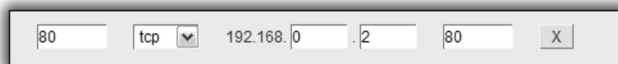


Figure 17.8 : *Paramétrage d'une redirection de port sur la Freebox*

Cette redirection indique que tous les accès sur le port `80` de la Freebox sont automatiquement redirigés vers le port `80` de votre machine de travail. Le port `80` est en effet le port par défaut utilisé par Apache et plus généralement pour le web.

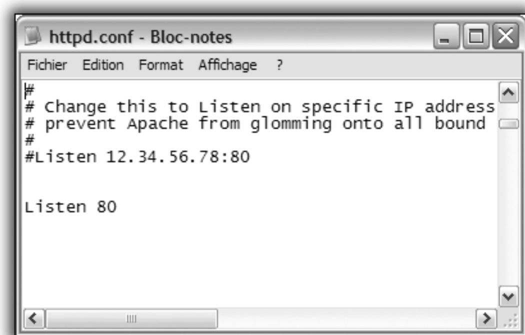


Figure 17.9 : *La configuration d'Apache indique bien que le port d'écoute est 80*

Cette modification doit être suivie d'un débranchement de la Freebox afin de la redémarrer. Tout est désormais prêt pour qu'un internaute lambda puisse accéder à votre site. La seule donnée manquante est l'adresse Internet de la Freebox. C'est en effet cette dernière qui permettra d'accéder au site. La rubrique *Afficher mon Adresse IP* permet de l'obtenir. Si cette adresse est `82.239.58.1` un internaute pourra utiliser l'adresse **http://82.239.58.1** pour visualiser le site.

Il est important de rappeler qu'en ouvrant le port `80` de cette manière, votre machine devient une proie potentielle pour un pirate. La prudence est donc de mise !

17.2. MySQL

Récupérer un enregistrement de manière aléatoire

Il est souvent très utile de récupérer une valeur aléatoire dans une table. La première solution consiste à sélectionner un grand nombre de valeurs, à les placer dans un tableau au niveau de votre script et, enfin, à choisir un élément au hasard dans ce tableau :

Listing 17-13 : Passage par un tableau pour obtenir un élément aléatoire

```
<?php

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");

$sql = "SELECT * FROM eleve";
$resultat = mysql_query ($sql);
while ($eleve = mysql_fetch_array ($resultat))
{
  $tab[] = $eleve['nom'];
}

srand ((float) microtime() * 10000000);
$randindex = array_rand ($tab);

echo $tab[$randindex];

mysql_close($liendb);

?>
```

Une autre solution, plus pertinente, consiste à demander à MySQL de ne retourner qu'une valeur prise au hasard. L'idée est de demander à MySQL de prendre les enregistrements dans un ordre aléatoire (`ORDER BY rand()`) et de n'en sélectionner qu'un (`LIMIT 1`) :

Listing 17-14 : Sélection directe d'un élément aléatoire

```
<?php

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");

$sql = "SELECT * FROM eleve ORDER BY rand() LIMIT 1";
$resultat = mysql_query ($sql);
$eleve = mysql_fetch_array ($resultat);
```

```
echo $eleve['nom'];

mysql_close($liendb);

?>
```



Anciennes versions de MySQL

Avec des versions anciennes de MySQL (inférieures à la version 3.23), vous ne pouvez utiliser que la première solution, la syntaxe `ORDER BY rand()` n'étant pas valide avant.

Optimiser ses tables

Il est courant, en cours de développement, d'apporter des modifications (`ALTER TABLE`) à la structure de ses tables. Bien que ces modifications soient parfaitement gérées par MySQL, elles peuvent entraîner un léger ralentissement des requêtes utilisant ces mêmes tables.

MySQL propose la commande `OPTIMIZE TABLE nom_table` pour restructurer ses données.

Cette commande peut aussi être utilisée sur des tables subissant des suppressions (massives) d'enregistrements. Dans ce cas, elle récupère l'espace perdu et défragmente le fichier de données.

Un appel à `OPTIMIZE TABLE` entraîne également un tri des pages d'index, ce qui ne fait jamais de mal.

Ces fonctions avancées (`REPAIR TABLE`, `CHECK TABLE`, `ANALYSE TABLE`, `OPTIMIZE TABLE`) peuvent être appelées depuis phpMyAdmin.

- Maintenance de la table : Vérifier la table [Documentation] - Analyser la table [Documentation]
  Réparer la table [Documentation] - Optimiser la table [Documentation]

Figure 17.10 : *Fonctions avancées dans phpMyAdmin*

Autres optimisations

Une table ne contenant que des colonnes de type numérique (`INT`, `TINYINT`, `FLOAT`…) et des colonnes de types `CHAR` ou `DATE` présente une structure dite statique. Cette structure lui permet d'être largement plus rapide sur la gestion des index et sur les recherches. La présence d'une seule colonne `TEXT`, `BLOB` ou `VARCHAR` la fera passer en structure dynamique. La taille d'un `CHAR` étant limitée à 255, il n'est pas toujours possible d'obtenir une structure statique de table.

Diviser une table en plusieurs sous-tables, sous prétexte que cette dernière commence à contenir beaucoup de colonnes, est généralement une mauvaise idée. La recherche de données sur le disque dur est en effet l'opération la plus longue. En divisant votre table, le SGBD est obligé de rechercher sur le disque les emplacements où se trouvent les sous-tables.

17.3. HTML et Javascript

Empêcher l'autocomplétion

Une fonctionnalité très pratique proposée par les derniers navigateurs est l'autocomplétion dans les formulaires (`autocomplete`). Cette autocomplétion permet à l'internaute de ne pas avoir à retaper une information qu'il a déjà précisée dans le passé.

Dans l'exemple suivant, l'internaute a entré une première fois son adresse e-mail : `bob@mail.com`. En revenant sur le site, cette adresse lui est directement proposée :

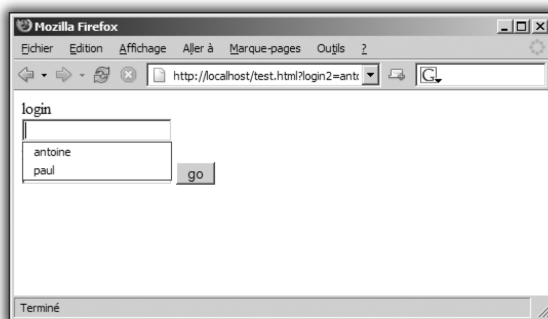


Figure 17.11 : *L'autocomplétion*

Ce comportement peut cependant être très gênant quand vous passez par un formulaire pour demander à l'internaute de s'identifier. Si l'ordinateur en question est accessible à d'autres personnes, ces dernières pourront d'identifier juste en tirant parti de l'autocomplétion. Il est donc important pour des champs « sensibles » de désactiver cette fonctionnalité. Il suffit, pour cela, de mettre la propriété `autocomplete` à `off` dans l'`INPUT TEXT` :

```
<input type="text" name="login"  autocomplete="off" />
```

Définir le rafraîchissement automatique d'une page

Il peut être utile d'avoir à rafraîchir votre page régulièrement (pour des pages de chat, d'informations en continue). HTML propose une balise `META` spéciale : `META HTTP-EQUIV="Refresh"`. Cette balise `META` permet de préciser la fréquence des rafraîchissements et l'URL de destination :

Listing 17-15 : Au bout d'une minute, nous retombons sur la page d'accueil

```
<META  HTTP-EQUIV="Refresh"  CONTENT="60;URL=/">
```

Le script suivant affiche l'heure toutes les 5 secondes :

```
<html>

<head>
<META HTTP-EQUIV="Refresh"
      CONTENT="5;URL=<?php echo $_SERVER['PHP_SELF']; ?>">
</head>

<body>

<h1><?php echo date("Y-m-d G:i:s"); ?></h1>

</body>
</html>
```



Problématique des frames

Il est impossible, en utilisant cette technique, de préciser une frame de destination. Pour y parvenir, vous êtes obligé de passer par une page intermédiaire contenant du Javascript.

Chapitre 18

Les fonctions PHP

Les fonctions mathématiques 529
Les chaînes de caractères 538
Les expressions régulières 559
Les tableaux 561
Les fonctions de dates et d'heures 583
Les fichiers et les répertoires 588
L'interface avec MySQL 611
Les images 622
Les variables 638
La configuration PHP 642
Fonctions diverses 645

Comme tous les nouveaux langages de haut niveau (C, C++, Java, C#, etc.), PHP est fourni par défaut avec un grand nombre de fonctionnalités. Cette bibliothèque de fonctions peut être assimilée à la librairie standard d'un langage de programmation (la *libc* pour le C, par exemple). Quel que soit l'endroit du script où vous vous trouvez, quel que soit l'interpréteur PHP utilisé, il vous est notamment possible d'appeler des fonctions de manipulation de nombres, de chaînes de caractères, de tableaux, etc.

En plus de ces fonctions fondamentales, il est également possible de faire appel à des fonctions plus « exotiques », qui permettent, par exemple, de gérer des bases de données, d'utiliser le protocole FTP, de générer des images ou des fichiers *.pdf*. Ce sont ainsi plusieurs dizaines d'extensions (aussi appelées « modules » ou « librairies ») qui sont venues se greffer à PHP au cours du temps. La fonction `phpinfo()` permet d'obtenir la liste des modules disponibles ainsi que leur version.

Figure 18.1 : *La distribution PHP contient notamment des modules de génération d'images (GD) et de gestion de protocole FTP*

Dans ce chapitre, nous nous intéresserons particulièrement aux fonctions permettant :

- le traitement des variables numériques ;
- le traitement des variables de type chaîne de caractères ;
- la manipulation des expressions régulières ;
- le traitement des tableaux et des tables de hachage (tableaux associatifs) ;
- la manipulation des dates et des heures ;
- la manipulation des fichiers et des répertoires ;
- la gestion des bases de données MySQL ;
- la génération d'images ;
- la manipulation des variables.

La liste des fonctions présentées ici n'est pas exhaustive, mais elle permet de répondre à 95 % des besoins d'un programmeur PHP.

Pour certaines fonctions, le nombre de paramètres peut être variable. Ces paramètres optionnels seront alors notés entre crochets de la manière suivante : `ma_fonction($param1[, $param2])`.

La fonction `ma_fonction()` peut prendre un ou deux paramètres. Une telle description d'une fonction est appelée « signature d'une fonction ».

Il est aussi possible de trouver des fonctions prenant par exemple un ou trois paramètres. La notation sera alors : `ma_fonction($param1[, $param2, $param3])`.

Nous préciserons également à partir de quelle version de PHP ces fonctions sont disponibles.

18.1. Les fonctions mathématiques

Il existe, au sein du langage PHP, des données particulières dont la valeur ne peut être modifiée dans le script : ces données sont appelées « constantes ». Il existe un certain nombre de constantes mathématiques qui peuvent être utilisées dans un script PHP.

Tableau 18.1 : Les constantes numériques		
Constante	**Valeur**	**Description**
M_PI	3.14159265358979323846	pi
M_E	2.7182818284590452354	e
M_LOG2E	1.4426950408889634074	log_2 e
M_LOG10E	0.43429448190325182765	log_10 e
M_LN2	0.69314718055994530942	log_e 2
M_LN10	2.30258509299404568402	log_e 10
M_PI_2	1.57079632679489661923	pi/2
M_PI_4	0.78539816339744830962	pi/4
M_1_PI	0.31830988618379067154	1/pi
M_2_PI	0.63661977236758134308	2/pi
M_SQRTPI	1.77245385090551602729	sqrt(pi) [4.0.2]
M_2_SQRTPI	1.12837916709551257390	2/sqrt(pi)
M_SQRT2	1.41421356237309504880	sqrt(2)
M_SQRT3	1.73205080756887729352	sqrt(3) [4.0.2]
M_SQRT1_2	0.70710678118654752440	1/sqrt(2)
M_LNPI	1.14472988584940017414	log_e(pi) [4.0.2]
M_EULER	0.57721566490153286061	Euler constant [4.0.2]

Exemple d'utilisation de constantes :

```
$x = M_E * 4;
print(M_PI);  // affiche 3.1415926535898
```

abs()

- Signature : `abs ($x).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la valeur absolue d'un nombre, quel qu'il soit :

```
print(abs(-12));  // affiche 12
print(abs(-2.12));  // affiche 2.12
```

base_convert()

- Signature : `base_convert ($x, $bo, $ba)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction convertit une valeur `$x` d'une base (`$bo`) dans une autre (`$ba`). Les valeurs de bases sont comprises entre 2 et 36 :

```
print(base_convert(13,10,2));
// conversion de la valeur 13 de la base décimale (10) dans
// la base binaire la valeur affichée est : 1101
print(base_convert("CC",16,10));
// affiche 204
// l'équivalent décimal de la couleur HTML #CCCCCC (gris)
// est donc (204,204,204)
```

bindec()

- Signature : `bindec ($ch)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction convertit en une valeur décimale une chaîne de caractères représentant une valeur codée en binaire. La plus grande valeur décimale que l'on puisse atteindre est 2 147 483 647 :

```
print(bindec("1101"));  // affiche 13
```

ceil()

- Signature : `ceil ($x)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne l'entier supérieur :

```
print(ceil(4.25));  // affiche 5
print(ceil(4.9));  // affiche 5
print(ceil(-2.45));  // affiche -2
```

cos()

- Signature : `cos ($x)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne le cosinus de la valeur `$x` (en radians) :

```
print(cos(M_PI));  // affiche -1
```

D'autres fonctions trigonométriques sont disponibles : `acos()`, `acosh()`, `asin()`, `asinh()` , `atan()`, `atanh()`, `atan2()`, `sin()`, `sinh()`, `tan()`, `tanh()`.

decbin()

- Signature : `decbin ($x)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la valeur convertie en binaire d'une valeur décimale `$x` :

```
print(decbin(2));
// affiche 10 en effet : 2 = 1 * 2^1 + 0 * 2^0
```

dechex()

- Signature : `dechex ($x)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la valeur convertie en hexadécimal (base 16) d'une valeur décimale `$x` :

```
print(dechex(204));  // affiche cc
```

decoct()

- Signature : `decoct ($x)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la valeur octale (base 8) d'une valeur décimale `$x` :

```
print(decoct(12));  // affiche 14
```

deg2rad()

- Signature : `deg2rad ($x)`.
- Versions : PHP 3 à partir de la 3.0.4, PHP 4, PHP 5.

Cette fonction convertit des degrés en radians :

```
print(deg2rad(45));  // affiche 0.78539816339745
```

L'inverse est réalisé par la fonction `rad2deg()` :

```
print(rad2deg(M_PI));  // affiche 180
```

exp()

- Signature : `exp ($n).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la puissance `$n` de la constante `e` :

```
print(exp(2));  // affiche 7.3890560989307
```

floor()

- Signature : `floor ($x).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne l'entier inférieur :

```
print(floor(4.25));  // affiche 4
print(floor(-2.45));  // affiche -3
```

getrandmax()

- Signature : `getrandmax ()`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la valeur aléatoire la plus grande qui puisse être générée sur le système qui interprète votre script :

```
print(getrandmax());
// affiche 2147483647 sur un PC sous Linux
```

Cette valeur dépend essentiellement de l'architecture de votre ordinateur (PC, Mac).

hexdec()

- Signature : `hexdec ($x).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction convertit une valeur hexadécimale en valeur décimale :

```
print(hexdec("A"));  // affiche 10
```

log()

- Signature : `log ($x)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne le logarithme d'une variable :

```
print(log(M_E));  // affiche 1
```

log10()

- Signature : `log10 ($x)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne le logarithme décimal d'une variable :

```
print(log10(M_E));  // affiche 0.43429448190325
```

max()

- Signature : `max ($x1 [, $x2…])`.
- Versions : PHP 3, PHP 4, PHP 5.

La fonction `max()` peut avoir deux types d'arguments.

- Un tableau : il retourne alors la valeur la plus grande du tableau.
- Deux nombres : il retourne alors le plus grand.

```
$tab = array (10,-2,4,2);
print(max($tab));  // affiche 10
```

```
print(max(3,5,2));  // affiche 5
```

Si une des valeurs au moins est un nombre flottant (`float`), la valeur retournée sera un nombre flottant.

La fonction inverse de `max()` est la fonction `min()` : elle fonctionne exactement sur le même modèle.

mt_rand()

- Signature : `mt_rand ([$min, $max])`.
- Versions : PHP 3 à partir de la 3.0.6, PHP 4, PHP 5.

Cette fonction permet de générer des nombres aléatoires de « meilleure qualité », et quatre fois plus rapidement qu'avec la fonction standard `rand()`. Si les valeurs optionnelles `$min` et `$max` ne sont pas précisées, la valeur aléatoire sera comprise entre `0` et `RAND_MAX`, sinon entre `$min` et `$max`.

La fonction `mt_srand()`, qui initialise le générateur de nombres aléatoires, doit être exécutée avant que `mt_rand()` soit utilisée :

```
mt_srand ((float) microtime() * 1000000);
// initialisation du générateur de nombres aléatoires

print(mt_rand(10,15));
// affiche, par exemple, 14
```

number_format()

- Signature : `number_format ($x [, $dec [, $chdec , $chmille]])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne une chaîne de caractères correspondant au nombre `$x` formaté suivant certains paramètres.

La fonction `number_format()` peut prendre plusieurs paramètres…

- Un paramètre : un chiffre. Il place alors une virgule entre les milliers, et les décimales ne sont pas prises en compte.

  ```
  print(number_format(1234567890));
  // affiche 1,234,567,890
  print(number_format(1234567890.234));
  // affiche 1,234,567,890
  ```

- Deux paramètres : un chiffre et le nombre de décimales à afficher. Il place alors une virgule entre les milliers du nombre, et un point devant les décimales.

  ```
  print(number_format(1234.987654321,5));
  // affiche 1,234.98765
  ```

- Quatre paramètres. Les deux derniers paramètres permettent de modifier les caractères de séparation des décimales et des milliers.

```
print(number_format(1234.987654321,5,"#","/"));
// affiche 1/234#98765
```

Cette fonction est extrêmement pratique pour afficher les nombres suivant les normes des pays :

```
$n = 12345.60;
// affichage à la française
print(number_format($n, 2, ',', ' '));
// affiche 12 345,60
// affichage à l'anglaise
print(number_format($n,2));
// affiche 12,345.60
```

octdec()

- Signature : `octdec ($x)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction convertit une valeur octale en valeur décimale :

```
print(octdec(17));  // affiche 15
```

pi()

- Signature : `pi ()`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la valeur de pi :

```
print(pi());  // affiche 3.1415926535898
```

pow()

- Signature : `pow ($x,$y)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la puissance `$y` du nombre `$x`. Les valeurs `$x` et `$y` peuvent être des nombres flottants :

```
print(pow(4,2));  // affiche 16
```

```
print(pow(4,0.5));
// affiche 2, il s'agit de la racine carrée
```

rand()

- Signature : `rand ([$min, $max])`.
- Versions : PHP 3, PHP 4, PHP 5.

Sans paramètre, cette fonction retourne une valeur aléatoire.

Avec deux paramètres, `$min` et `$max`, elle retourne une valeur aléatoire comprise entre ces deux valeurs.

Avant d'utiliser cette fonction, il est nécessaire d'utiliser la fonction `srand()` afin d'initialiser le générateur de nombres aléatoires :

```
srand ((double) microtime() * 1000000);
print(rand());
// affiche une valeur aléatoire
print(rand(3,15));
// affiche une valeur aléatoire entre 3 et 15
```

round()

- Signature : `round ($x [, $precision])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction arrondit un nombre flottant. La précision peut être passée en paramètre :

```
print(round("10.1"));  // affiche : 10
print(round("10.6"));  // affiche : 11
print(round("10.129",2));  // affiche : 10.13
```

srand()

- Signature : `srand ($seed)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet d'initialiser le générateur de nombres aléatoires. Pour que le hasard soit maximal, il est préférable de n'appeler cette fonction qu'une fois dans le script.

La valeur `(double)microtime()*1000000` est très courante pour le paramètre `$seed`.

Voyez aussi la fonction `rand()` pour la génération effective de nombres aléatoires, et les fonctions `mt_rand()` et `mt_srand()` comme des alternatives supérieures à `rand()` et `srand()`.

sqrt()

- Signature : `sqrt ($x)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la racine carrée d'un nombre :

```
print(sqrt(121));  // affiche : 11
```

18.2. Les chaînes de caractères

Nous allons nous intéresser, dans cette partie, aux fonctions concernant les chaînes de caractères. En informatique, une chaîne de caractères est généralement appelée une *string*. Nous utiliserons donc régulièrement le paramètre `$str` quand nous voudrons faire référence à une variable de type chaîne de caractères.

addslashes()

- Signature : `addslashes ($str)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction ajoute des barres obliques inversées (*antislashs*) devant les caractères (`'`, `"`, `\`, `NUL`) de la chaîne `$str` :

```
print(addslashes("bon'jour\"monde"));
// affiche  bon\'jour\"monde
```

bin2hex()

- Signature : `bin2hex ($str)`.
- Versions : PHP 3 à partir de la 3.0.9, PHP 4, PHP 5.

Cette fonction convertit des données binaires dans leur représentation hexadécimale (le bit de poids fort en premier).

chop()

- Signature : `chop ($str)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la chaîne sans les blancs de fin de ligne. On regroupe, parmi les caractères dits blancs : les espaces, les tabulations (`\t`), les sauts de ligne (`\n`), les retours chariot (`\n \r`) et le caractère nul (`\0`) :

```
$str = " bonjour monde    ";

print("|" . $str . "|");
// affiche | bonjour monde    |

print("|" . chop($str) . "|");
// affiche | bonjour monde|
```

chr()

- Signature : `chr ($n)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne le caractère ayant pour code ASCII `$n` :

```
print(chr(169));  // affiche : ©
```

chunk_split()

- Signature : `chunk_split ($str [, $n [, $sep ]])`.
- Versions : PHP 3 à partir de la 3.0.6, PHP 4, PHP 5.

Cette fonction découpe une chaîne `$str` en plusieurs morceaux, de taille `$n`, séparés par la chaîne `$sep` :

```
print(chunk_split("abcde",2,"--<br>"));
/* affiche :
ab--
cd--
e--
*/
```

```
print(chunk_split("0921310571",2);
// affiche : 09 21 31 05 71
// pratique pour afficher un numéro de téléphone
```

Si seul le paramètre `$str` est fourni, `$n` a pour valeur par défaut `76` et `$sep` (`\r\n`).

À titre d'information, la valeur `76` représente le nombre maximal de caractères conseillé par ligne pour le corps d'un e-mail.

count_chars()

- Signature : `count_chars ($str, $mode)`.
- Versions : PHP 4, PHP 5.

Cette fonction permet d'obtenir des informations sur les caractères contenus dans `$sr`.

Les modes 1 et 3 sont particulièrement intéressants.

- Mode 1 : la fonction retourne un tableau avec comme clé le code ASCII du caractère et comme valeur le nombre d'occurrences de ce caractère dans `$str`.

```
$tab = count_chars("azeerrrty",1);
while (list ($clef, $val) = each ($tab))
{
  print("$clef : " . chr($clef) . " : [$val]<br>");
}
/* affiche :
97 : a : [1]
101 : e : [2]
114 : r : [3]
116 : t : [1]
121 : y : [1]
122 : z : [1]
*/
```

- Mode 3 : la fonction retourne une chaîne de caractères contenant tous les caractères différents présents dans `$str`.

```
print(count_chars("azeerrrty",3));
// affiche : aertyz
```

crc32()

- Signature : `crc32 ($str)`.
- Versions : PHP 4, PHP5.

Cette fonction calcule le code à redondance cyclique d'une chaîne.

À titre d'information, les valeurs utilisées dans l'algorithme sont `0xEDB88320` et `0xFFFFFFFF`.

crypt()

- Signature : `crypt ($str [, $salt ])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction crypte la chaîne `$str` en utilisant l'algorithme DES.



La fonction `uncrypt()`

Cette fonction est à sens unique. La fonction `uncrypt()` n'existe pas (et n'est pas près d'exister !).

echo()

- Signature : `echo ($str1 [, $str2…])`.

Cette fonction permet d'afficher une ou plusieurs chaînes de caractères :

```
$a = "toto";
$b = "titi";
echo $a, "-", "$b";  // affiche : toto-titi
echo ($i > 5) ? "i supérieur à 5" : "i inférieur ou égal à 5";
// affiche : i inférieur ou égal à 5
```



Le paramètre `echo` **simplifié**

Les deux lignes suivantes sont équivalentes :

```
bonjour <?php echo "monde"; ?>
bonjour <?="monde"?>
```

explode()

- Signature : `explode ($sep, $str [, $limite])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne un tableau contenant les différentes parties de la chaîne de caractères `$str` séparées par la chaîne de caractères `$sep` :

```
$tab = explode("_","bonjour_monde");
print($tab[1]);  // affiche : monde
$tab = explode(";;","a;;b;;c;;;d");
print_r($tab);
/* affiche
Array
(
    [0] => a
    [1] => b
    [2] => c
    [3] => ;d
)
*/
```

Si le paramètre `$limite` est transmis, le tableau contiendra le plus grand nombre d'éléments `$limite`, le dernier contenant le reste de la chaîne :

```
$tab = explode(";;","a;;b;;c;;;d",2);
print_r($tab);
/* affiche
Array
(
    [0] => a
    [1] => b;;c;;;d
)
*/
```

get_meta_tags()

- Signature : `get_meta_tags ($fichier)`.
- Versions : PHP 3 à partir de la 3.0.4, PHP 4, PHP 5.

Cette fonction retourne un tableau associatif contenant les métabalises du fichier `$fichier`. La variable `$fichier` peut contenir une URL :

```
$tab = get_meta_tags ("http://www.lemonde.fr");
foreach ($tab as $nom => $valeur)
{
  print("$nom => $valeur<br>");
}
/* affiche
robots => INDEX,FOLLOW,NOARCHIVE
description => LE MONDE, Journal Le Monde, quotidien
d'information francophone / Le Monde, the french quality
newspaper of record
```

```
keywords => LE MONDE, INFORMATIONS, INFOS, QUOTIDIEN,
DAILY NEWS, PRESSE, PRESS, etc.
*/
```

htmlentities()

- Signature : `htmlentities ($str [, $mode ])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction convertit certains caractères spéciaux d'une chaîne qui ont un équivalent en HTML. Ainsi, le caractère `&` peut être représenté en HTML par `&` :

```
$str = "< bonjour c'est l'été & il fait très beau >";
print(htmlentities($str));
```

Cet exemple affiche bien `"< bonjour c'est l'été & il fait très beau >"`. Cependant, en regardant les sources de la page, on s'aperçoit que la chaîne de caractères est devenue :

```
"< bonjour c'est l'été & il fait très beau >"
```

Il est possible de passer un deuxième paramètre pour préciser comment les guillemets doivent être traités. Ce paramètre est une des constantes suivantes…

- `ENT_COMPAT` : seuls les guillemets sont convertis en `'`.
- `ENT_QUOTES` : primes et guillemets sont convertis.
- `ENT_NOQUOTES` : aucune conversion des guillemets n'est réalisée.

Cette fonction peut être très utile si vous voulez afficher un texte qui soit compatible avec le plus grand nombre de navigateurs. En effet, les vieilles générations de navigateurs avaient, par exemple, des problèmes pour afficher un caractère accentué qui n'avait pas été converti.

Il existe une fonction similaire, `htmlspecialchars()`, qui ne convertit qu'un certain nombre de caractères : `&`, `'`, ", <, >.

implode()

- Signature : `implode ($s, $tab)`.
- Versions : PHP 3, PHP 4, PHP 5.

Fonction inverse de `explode()`, elle permet de réunir tous les éléments d'un tableau dans une chaîne en les joignant avec la chaîne de caractères `$s` :

```
$tab = array("www","kernix","com");
print(implode(".",$tab));  // affiche www.kernix.com
```

Il existe une fonction parfaitement identique à `implode()` : `join()`.

md5()

- Signature : `md5 ($str)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction crypte la chaîne `$str` en utilisant l'algorithme MD5.

nl2br()

- Signature : `nl2br ($str)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction convertit les sauts de ligne texte d'une chaîne en saut de ligne HTML : `<br/>`. Il s'agit de `<br/>` et non de `<br>`, car le Web va de plus en plus s'orienter vers le langage XHTML (plus proche du XML) où les balises « orphelines » doivent être « fermées ».



Balises orphelines

On entend par « balises orphelines », les balises qui n'ont pas besoin d'être fermées : `<hr> <input> <img>`).

```
print(nl2br("bonjour\nmonde"));
/* affiche :
bonjour<br />
monde
*/
```

ord()

- Signature : `ord ($str)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la valeur ASCII du premier caractère de la chaîne de caractères `$str` :

```
print(ord("(c) copyright FCB"));  // affiche : 169
```

parse_str()

- Signature : `parse_str ($str [, $tab])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction crée les variables contenues dans la *query string* `$str` et les initialise à leur valeur :

```
$str = "val1=bonjour+monde&val2=coucou";
parse_str($str);
echo $val1, " - ", $val2;
// affiche : bonjour monde - coucou
```

Il est aussi possible de passer un tableau en deuxième paramètre :

```
$str = "val1=bonjour+monde&val2=coucou";
parse_str($str,$tab);
print($tab["val2"]);  // affiche : coucou
```

print()

- Signature : `print ($str)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction affiche la variable :

```
$v = 'bonjour';
print("$v monde");  // affiche : bonjour monde
print "bonjour";  // affiche : bonjour
```

printf()

- Signature : `printf ($format [, $param1, $param2…])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet d'afficher une chaîne de caractères « mise en forme ».

Le premier paramètre correspond à la chaîne de caractères de mise en forme : le format. Il peut être suivi d'un ou de plusieurs paramètres qui seront les données utilisées lors de l'affichage :

```
$nom = "Fred";
printf("Bonjour %s",$nom);  // affiche : Bonjour Fred
```

La chaîne `"Bonjour %s"` représente le format et `%s` indique qu'une chaîne de caractères va être placée à cet endroit. Il existe d'autres séquences de formatage que `%s` :

- `%%` affiche un %.
- `%c` affiche la valeur ASCII d'un nombre.
- `%d` affiche un entier.
- `%f` affiche une valeur décimale à virgule.
- `%o` affiche une valeur décimale en octal.
- `%b` affiche une valeur décimale en binaire.
- `%x` affiche une valeur décimale en hexadécimal (les lettres sont en minuscules).
- `%X` affiche une valeur décimale en hexadécimal (les lettres sont en majuscules).

```
$nom = "Bob";
$annee = 2001;
$taux = "12.5";

printf("%c %d %d, %s possède %f %% du
✂ capital",169,2000,$annee,$nom,$taux);
// affiche : © 2000 2001, Bob possède 12.500000 % du
✂ capital

printf("décimal[%d] - binaire [%b] - octal[%o] -
✂ hexadécimal[%x]",29,29,29,29);
// affiche : décimal[29] - binaire [11101] -
✂ octal[35]
//           hexadécimal[1d]
```

Les paramètres suivant la chaîne de formatage (`$format`) doivent donc être dans le même ordre que les séquences de formatage.

Vous vous apercevez, dans le premier exemple, que la valeur `12.5` est affichée avec cinq zéros. Il s'avère que la séquence de formatage `%f` peut être modifiée pour afficher un nombre exactement comme on le souhaite : `%.2f` va, par exemple, afficher un nombre flottant avec deux chiffres après la virgule.

```
printf("prix en francs : %.2f FF, soldé à %.2f
✂ FF",132.569,112.1);
// affiche : prix en francs : 132.57 FF, soldé à 112.10 FF
```

S'il y a lieu, les décimales sont complétées avec des `0`.

Il est aussi possible de formater un nombre sur une certaine longueur :

```
printf("référence %05d",13);  // affiche : référence 00013
```

La fonction `sprintf()` est similaire à `printf()`, sauf qu'elle n'affiche pas le résultat, mais retourne la chaîne formatée.

```
$val = sprintf("%.2f",12.1);
echo "prix = $val FF";  // affiche : prix = 12.100 FF
```

quotemeta()

- Signature : `quotemeta ($str).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction ajoute des barres obliques inversées devant les caractères suivants : `.`, `\`, `+`, `*`, `?`, `[`, `^`, `]`, `(`, `$`, `)`.

```
print(quotemeta("+*("));  // affiche : \+\*\(
```

sscanf()

- Signature : `sscanf ($str, $format, [$var1, $var2…]).`
- Versions : PHP 4 à partir de la 4.0.1, PHP 5.

Cette fonction permet de lire une chaîne de caractères et d'en extraire les différentes valeurs grâce à une chaîne de formatage similaire à celle de `printf()` :

```
$naissance = "11/03/1977";
$tab = sscanf($naissance,"%d/%d/%d");
print("année de la naissance : " . $tab[2]);
// affiche : 1977
$ligne = "PRIX|2134.3000 REFERENCE|bzr0145";
list ($prix,$ref) = sscanf($ligne,"PRIX|%f REFERENCE|%s");
print("prix du produit [$ref] : $prix FF");
// affiche : prix du produit [bzr0145] : 2134.3 FF
```

Si les paramètres `$var1`, `$var2`, etc., sont transmis, la fonction assigne ces variables et retourne le nombre de variables qui ont été assignées. Ces paramètres optionnels doivent être passés par référence :

```
$naissance = "11/03/1977";
echo sscanf($naissance,"%d/%d/%d",&$jour,&$mois,&$annee);
// affiche 3
echo "année = $annee";
// affiche : année = 1977
```

strcmp()

- Signature : `strcmp ($str1, $str2)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction compare deux chaînes de caractères et retourne :

- une valeur négative si `$str1` est inférieure à `$str2` ;
- une valeur positive si `$str1` est supérieure à `$str2` ;
- `0` si elles sont égales.

Attention, ce ne sont pas les longueurs des chaînes qui sont comparées, mais les caractères qu'elles contiennent ! Si vous voulez comparer les chaînes `"abcxwtr"` et `"ac"`, l'algorithme effectue ceci :

- Premier caractère : `a = a`.
- Deuxième caractère : `c > b`.

L'algorithme s'arrête alors et déclare que la chaîne `"ac"` est supérieure à `"abcxwtr"`.

Les comparaisons se font donc sur les valeurs (codes) ASCII des caractères. Or, dans la table ASCII, l'ordre est établi ainsi : 0 < 9 < A < Z < a < z.

```
$str1 = "aa";
$str2 = "b";
$n = strcmp($str1,$str2);
if ($n < 0) print("str1 < str2");
elseif ($n > 0) print("str1 > str2");
else print("str1 = str2");
// affiche : str1 < str2
```

Si seule l'égalité vous intéresse, il est souvent plus simple d'utiliser l'opérateur de comparaison == :

```
$str1 = "hello";
if ($str1 == "hello") print("OK");
else print("KO");
// affiche : OK
```

Dans les deux exemples précédents, les chaînes `"bonjour"` et `"BonJour"` ne seraient pas identiques, car les comparaisons sont sensibles à la différence majuscules/minuscules (les comparaisons sont dites *case sensitive*).

La fonction `strcasecmp()` peut être utilisée si vous ne voulez pas vous soucier de la différence majuscules/minuscules :

```
$str1 = "HELLO";
if (strcasecmp($str1,"hello") == 0) print("OK");
else print("KO");
// affiche : OK
```

Il est aussi possible de limiter les portions de chaînes à comparer avec les fonctions `strncmp()` ou `strncasecmp()` :

```
if (strncasecmp("HELIOS","hello",3) == 0) print("OK");
else print("KO");
// affiche : OK
```

La comparaison s'est faite ici sur les trois premiers caractères et de manière *insensitive*.

Avec ces types de fonctions, la chaîne `"image2"` est supérieure à `"image10"`. Or, il est souvent utile pour classer des images ou des fichiers de faire en sorte d'avoir `"image1"` < `"image2"` < `"image12"` < `"image21"`. Il faut, pour cela, utiliser un algorithme de comparaison dit naturel. Deux fonctions implémentent cet algorithme : `strnatcmp()` et `strnatcasecmp()`.

Pour classer un tableau en utilisant cet ordre, la fonction `natsort()` peut être utilisée.

strcspn()

- Signature : `strcspn ($str1, $str2)`.
- Versions : PHP 3 à partir de la 3.0.3, PHP 4, PHP 5.

Cette fonction retourne la longueur de la partie initiale de `$str1` qui ne contient aucun caractère de `$str2` :

```
print(strcspn ("aeiouy", "bdcdf"));  // affiche 6
print(strcspn ("aeiouy", "bdcdfi"));  // affiche 2
```

strip_tags()

- Signature : `strip_tags($str [, $balises])`.
- Versions : PHP 3 à partir de la 3.0.8, PHP 4 à partir de la 4.0b2, PHP 5.

Cette fonction retourne une chaîne privée des balises HTML et PHP :

```
$data = "Bonjour <b>monde</b><br>hello <font>world</font>";
$str = strip_tags($data);
// la variable $str contient alors "Bonjour mondehello world"
```

C’est très pratique quand vous voulez empêcher un internaute d’inclure des balises HTML dans des données qui seront stockées dans votre base de données.

Il est cependant possible d’autoriser certaines balises en transmettant un deuxième paramètre à la fonction `strip_tags()`. On peut, par exemple, n’autoriser que la mise en gras (<b>) et les sauts de ligne (
) :

```
$data = "Bonjour <b>monde</b><br>hello <font>world</font>";
$str = strip_tags($data,"<b><br>");
// la variable $str contient alors "Bonjour <b>monde</b>
// <br>hello world"
```

stripslashes()

- Signature : `stripslashes ($str)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction réalise l’opération inverse de la fonction `addslashes()` ; par exemple, `\’` devient `’` :

```
print(stripslashes("l\’aventure"));
// affiche : l’aventure
```

strlen()

- Signature : `strlen ($str).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la longueur d'une chaîne de caractères :

```
$password = "jkw6d";
if (strlen($password) < 8)
 print("le mot de passe doit contenir au moins 8
 ✂ caractères");
else
 print("longueur valide");
// affiche : le mot de passe doit contenir au moins 8
✂ caractères
```

str_pad()

- Signature : `str_pad ($str, $ln [,$s ,$mode]).`
- Versions : PHP 4 à partir de la 4.0.1, PHP 5.

Cette fonction permet de formater une chaîne de caractères à l'intérieur d'une plus grande chaîne, en précisant la longueur, l'alignement (gauche, milieu, droit) et les caractères de remplissage (s'il y a lieu) :

```
$str = "bonjour";
print(str_pad($str, 10));
// affiche : "bonjour   "
print(str_pad($str, 10, "-+", STR_PAD_LEFT));
// affiche : "-+-bonjour"
print(str_pad($str, 10, "_", STR_PAD_BOTH));
// affiche : "_bonjour__"
print(str_pad($str, 10, ".", STR_PAD_RIGHT));
// affiche : "bonjour..."
```

`STR_PAD_RIGHT` est le mode d'alignement par défaut, il est donc souvent inutile de le spécifier.

strpbrk()

- Signature : `strpbrk ($str, $list).`
- Version : PHP 5.

Cette fonction retourne une partie de `$str` commençant par l'un des caractères présents dans `$list`.

```
$str = "Bonjour Jean Dupont";
print(strpbrk($str,"ijq"));
// affiche "jour Jean Dupont" car le caractère 'i' n'est pas
// trouvé à la différence de 'j'
print(strpbrk($str,"iJq"));
// affiche " Jean Dupont" car cette fonction est sensible à
// la casse
```

strpos()

- Signature : `strpos ($str1, $str2)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la position de la variable `$str2` dans la variable `$str2` :

```
$str1 = "bonjour";
$str2 = "on";
print(strpos($str1,$str2));  // affiche : 1
```

Si `$str2` n'est pas présente dans `$str1`, la valeur booléenne `false` est retournée, ce qui peut conduire à une erreur assez courante :

```
$str1 = "bonjour";
$str2 = "bon";
if (strpos($str1,$str2) == false)
  print("$str2 n'a pas été trouvé dans $str1");
else
  print("OK");
// ce code affiche "bon n'a pas été trouvé dans bonjour"
```

Le résultat de cet exemple est donc erroné. Le problème vient du test ==, qui ne fait pas la différence entre les valeurs `false` et `0`. Il est donc nécessaire d'utiliser l'opérateur ===, qui compare à la fois la valeur et le type. La valeur `false` étant un booléen et `0` un numérique, le test ne passera pas :

```
$str1 = "bonjour";
$str2 = "bon";
if (strpos($str1,$str2) === false)
  print("$str2 n'a pas été trouvé dans $str1");
else
  print("OK");
// affiche : OK
```

La fonction `strrpos()` retourne, quant à elle, la dernière position de la variable `$str2` dans `$str1` :

```
echo strrpos("bonbon","on");  // affiche : 4
```

strrchr()

- Signature : `strrchr($str, $c).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la partie de `$str` qui commence là où l'on trouve pour la dernière fois la première lettre de la chaîne de caractères `$c`. En cas d'erreur, la valeur `false` est retournée :

```
$str = "www.toto.fr";
print("extension : " . strrchr($str,"."));
// affiche : extension : .fr
```

str_repeat()

- Signature : `str_repeat ($str, $n).`
- Versions : PHP 4 à partir de la 4.0b4, PHP 5.

Cette fonction répète une chaîne plusieurs fois :

```
print(str_repeat ("--", 4));
// affiche : --------
```

strrev()

- Signature : `strrev ($str).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction inverse une chaîne de caractères :

```
print(strrev("abc"));
// affiche : "cba"
```

strrpos()

- Signature : `strrpos ($str1, $str2).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la position de la dernière occurrence de la chaîne `$str2` au sein de `$str1` :

```
print(strrpos("hellohello","lo"));
// affiche : 8
```

strspn()

- Signature : `strspn ($str1, $str2).`
- Versions : PHP 3 à partir de la 3.0.3, PHP 4, PHP 5.

Cette fonction retourne la longueur du segment initial de `$str1`, qui n'est composé que de caractères de `$str2` :

```
print(strspn("bonjour","bonsoir"));
// affiche : 3
print(strspn("bonjour","on"));
// affiche : 0, car "on" n'est pas au début de "bonjour"
```

strstr()

- Signature : `strstr ($str1, $str2).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la partie de `$str1` qui commence là où l'on trouve pour la première fois la chaîne de caractères `$str2`. En cas d'erreur, le booléen `false` est retourné :

```
$email = "bob@toto.fr";
print("domaine : " . strstr($email,"@"));
// affiche : domaine : @toto.fr
```

La fonction `strchr()` est un alias de `strstr()`.

La fonction `stristr()` a le même comportement que `strstr()`, sans être sensible à la casse.

strtolower()

- Signature : `strtolower ($str).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la chaîne convertie entièrement en minuscules :

```
$str = "Bonjour MONDE";
print(strtolower($str));
// affiche : "bonjour monde"
```

Il existe d'autres fonctions permettant de jouer sur les minuscules et les majuscules.

- `strtoupper()` : la chaîne passe en majuscules.
- `ucfisrst()` : le premier caractère de la chaîne passe en majuscule.
- `ucwords()` : le premier caractère de chacun des mots passe en majuscule.

```
$str = "bonjour monde";
print(strtoupper($str));  // affiche : BONJOUR MONDE
print(ucfirst($str));  // affiche : Bonjour monde
print(ucwords($str));  // affiche : Bonjour Monde
```

str_replace()

- Signature : `str_replace ($str1,$str2,$str).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de remplacer la chaîne de caractères `$str1` par la chaîne de caractères `$str2` dans la chaîne de caractères `$str` :

```
print(str_replace("#PRENOM#","Paul","Prénom : #PRENOM#"));
// affiche : Prénom : Paul
```

Il est aussi possible de remplacer `$str1` et `$str2` par des tableaux :

```
$s = str_replace(array("#PRENOM#","#NOM#"),
                 array("Paul","Dupont"),
                 "Nom : #NOM#<br>Prénom : #PRENOM#");
print($s);
// affiche :
// Nom : Dupont
// Prénom : Paul
```

Cette fonction peut être très utile pour travailler avec des pages modèles (*templates*).

str_split()

- Signature : `str_split ($str,$n).`
- Version : PHP 5.

Cette fonction convertit la chaîne `$str` en un tableau. Le tableau est composé de parties de chaînes de longueur `$n`.

```
$tab = str_plit ("bonjour");
// $tab contient "b","o","n","j","o","u","r"
$tab = str_plit ("bonjour",4);
// $tab contient "bonj","our"
```

strtr()

- Signature : `strtr ($str, $chars1, $chars2).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de substituer, dans la chaîne `$str`, les caractères de `$chars1` par les caractères de `$chars2` :

```
$str = "cet élève est bête";
print(strtr($str, "éèêë", "eeee"));
// affiche : "cet eleve est bete"
```

substr()

- Signature : `substr ($str, $deb [, $long]).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet d'extraire une partie d'une chaîne de caractères.

Le premier paramètre est la chaîne de caractères en question, le deuxième la position à partir de laquelle l'extraction va être faite, le troisième (optionnel) indique la longueur de la partie à extraire.

Si le deuxième paramètre est négatif, la position est à considérer à partir de la fin de la chaîne.

Si le troisième paramètre est négatif (–`n`), la partie extraite s'arrêtera `n` caractères avant la fin de la chaîne :

```
// toute la chaîne sauf le premier caractère
print  substr("abcdef", 1);  // affiche : "bcdef"
```

```
// une chaîne de 3 caractères à partir du deuxième caractère
print  substr("abcdef", 1, 3);  // affiche : "bcd"

// les 2 derniers caractères
print  substr("abcdef", -2);  // affiche : "ef"

// toute la chaîne sauf les 2 derniers caractères
print  substr("abcdef", 0, -2);  // affiche : "abcd"

// toute la chaîne sauf le premier et le dernier caractère
print  substr("abcdef", 1, -1);  // affiche : "bcde"
```

substr_compare()

- Signature : `substr_compare ($str1,$str2, $deb [, $long [, $nocase]])`.
- Version : PHP 5.

Cette fonction permet de comparer la chaîne `$str1` (à partir de la position `$deb`) avec `$str2`, sur une longueur (optionnelle) de `$long` caractères. Si le paramètre `$nocase` est précisé et qu'il est égal à `true`, la fonction devient insensible à la casse.

```
echo substr_compare("abcde", "bc", 1, 2); // 0
echo substr_compare("abcde", "bcg", 1, 2); // 0
echo substr_compare("abcde", "BC", 1, 2, true); // 0
echo substr_compare("abcde", "bc", 1, 3); // 1
```

substr_count()

- Signature : `substr_count ($str1,$str2)`.
- Versions : PHP 4 à partir de la 4.0RC2, PHP 5.

Cette fonction permet de compter le nombre de fois où apparaît une chaîne (`$str2`) dans une autre (`$str1`) :

```
$n = substr_count("bonjour monde","on");
print $n;  // affiche : 2
```

substr_replace()

- Signature : `substr_replace ($str1, $str2, $deb [, $long] )`.

- Versions : PHP 4 à partir de la 4.0b4, PHP 5.

Cette fonction remplace une partie d'une chaîne de caractères `$str1` par une autre `$str2`.

Une position de départ (`$deb`) et une longueur optionnelle (`$long`) peuvent être précisées. Le mode de fonctionnement est alors le même que pour la fonction `substr()` :

```
$str = "bonjour monde";
// remplace $str par "--"
print substr_replace ($str, "--", 0);  // affiche : "--"
// insère "--" au début de $str
print substr_replace ($str, "--", 0, 0);
// affiche : "--bonjour monde"
// remplace la deuxième lettre de $str par "--"
print substr_replace ($str, "--", 1, 1);
// affiche : "b--njour monde"
// remplace les 4 premiers caractères de $str par "--"
print substr_replace ($str, "--", 0, 4);
// affiche : "--our monde"
// remplace la dernière de $str lettre par "--"
print substr_replace ($str, "--", -1, 1);
// affiche : "bonjour mond--"
// remplace par "--" tous les caractères de $str
// sauf les 2 derniers
print substr_replace ($str, "--", 0, -2);
// affiche : "--de"
// remplace par "-" tous les caractères de $str
// sauf le premier et le dernier
print substr_replace ($str, "--", 1, -1);
// affiche : "b--e"
```

trim()

- Signature : `trim ($str)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction supprime les caractères blancs de début et de fin de chaîne. Les caractères blancs sont les mêmes que pour la fonction `chop()` : `\n`, `\r`, `\t`, `\0`.

```
$str = " bonjour monde ";

print("|". $str . "|");
// affiche : | bonjour monde |
```

```
print("|". trim($str) . "|");
// affiche : |bonjour monde|
```

Il existe des variantes à `trim()` :

- `ltrim()` ne supprime que les caractères blancs à gauche.
- `rtrim()` ne supprime que les caractères blancs à droite (comme la fonction `chop()`).

wordwrap()

- Signature : `wordwrap ($str, [, $long [, $sep [, $coup]]])`.
- Versions : PHP 4 à partir de la 4.0.2, PHP 5.

Cette fonction permet de séparer une chaîne de caractères `$str` en plusieurs parties de longueur `$long`, séparées par une chaîne de caractères `$sep`.

Par défaut, la valeur de `$long` est `75` et celle de `$sep` est `\n`.

Par défaut, `wordwrap()` garde les mots en entier, sauf si `$coup` vaut `1` :

```
$str = "bonjour à tous";

print wordwrap($str,4,"<br>");
/* affiche :
bonjour
à
tous
*/

$str = "bonjour à tous";
print wordwrap($str,4,"<br>",1);
/* affiche :
bonj
our
à
tous
*/
```

18.3. Les expressions régulières

Dans cette partie, le paramètre `$motif` correspond à la `REGEXP`.

preg_match()

- Signature : `preg_match ($motif, $str)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de tester si le motif `$motif` est présent dans la chaîne `$str`. Elle retourne la valeur booléenne `true` si le motif est trouvé, sinon elle renvoie `false` :

```
if (preg_match("/important/",$str))
// permet de tester si $str contient le mot "important"
```

Le motif peut contenir des expressions régulières :

```
if (preg_match("/^\d*$/", $str))
// permet de tester si $str ne contient que des chiffres
```

preg_replace()

- Signature : `preg_replace ($motif, $modele, $str)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction balaie la chaîne `$str` pour trouver des zones de correspondance avec le motif `$motif` et remplace les zones en correspondance avec la chaîne `$modele` :

```
print(preg_replace("/c/","aaa","cqsdc"));
// affiche : aaaqsdaaa
$str = preg_replace("/\r\n/","\n",$str);
// permet de remplacer \r\n par \n dans la chaîne $str
print(preg_replace("/\d/","X","tel : 01 23 45 56 78"));
// tous les caractères de type numériques
// sont remplacés par 'X'
// le script affiche donc : tel : XX XX XX XX XX
```

split()

- Signature : `split ($motif, $str [, $limite])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de découper une chaîne à partir d'un motif de séparation. Les éléments issus de la « découpe » sont placés dans un tableau :

```
$tab = split("[ \.]","03.21.52.68.41");
// permet de découper un numéro de téléphone
// le caractère de séparation pouvant être un espace
// ou un point
print "indicatif : $tab[0]";  // indicatif : 03
```

Si le paramètre optionnel `$limite` est précisé, le tableau contiendra un grand nombre de `$limite` éléments, le dernier élément contenant la chaîne entière.

Si le motif `$motif` est un simple caractère (ou une simple chaîne), il est préférable d'utiliser la fonction `explode()` :

```
$tab = explode(",",$str);
```

18.4. Les tableaux

Les fonctions suivantes peuvent également être intéressantes dans le cadre de l'utilisation de tableaux : `is_array()`, `explode()`, `implode()`, `split()` et `unset()`.

array()

- Signature: `array()`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de créer un tableau.

Il est possible de créer :

- des tableaux scalaires ;

  ```
  $tab = array("rouge","jaune","bleu");
  print $tab[1];  // affiche : "jaune"
  ```

- des tableaux associatifs.

  ```
  $tab = array("nom"=>"dupont","prenom"=>"paul");
  print $tab["nom"];  // affiche : "dupont"
  ```

Il est aussi possible de créer des tableaux à plusieurs dimensions :

```
$toile = array (
"dimensions" => array ("longueur"=>"160", "largeur"=>"400"),
"peintre" => array ("prenom"=>"paul", "nom"=>"cezanne"),
);
print $toile["peintre"]["nom"];  // affiche : "cezanne"
```

array_combine()

- Signature: `array_combine ($tabindices, $tabvaleurs)`.
- Version : PHP 5.

Cette fonction retourne un tableau dont les indices sont les éléments de `$tabindices` et les valeurs, les éléments de `$tabvaleurs`.

Elle retourne `FALSE` si le nombre d'éléments ne correspond pas entre `$tabindices` et `$tabvaleurs`.

```
$tab1 = array ("a", "b", "c");
$tab2 = array ("x","y","z");
$tab = array_combine($tab1, $tab2);
// $tab contient : Array ( [a] => x [b] => y [c] => z )
```

array_count_values()

- Signature: `array_count_values ($tab)`.
- Versions : PHP 4 à partir de la 4.0b4, PHP 5.

Cette fonction retourne un tableau dont les indices sont les éléments du tableau `$tab` et dont les valeurs correspondent à la fréquence d'apparition de ces éléments dans `$tab`.

```
$tab = array ("bonjour","monde","bonjour");
$tab2 = array_count_values ($array);
print $tab2["bonjour"];  // affiche : 2
print $tab2["monde"];  // affiche : 1
```

array_diff()

- Signature : `array_diff($tab1,$tab2)`.
- Versions : PHP 4 à partir de la 4.0.1, PHP 5.

Cette fonction retourne un tableau qui contient toutes les valeurs de `$tab1` qui ne sont pas présentes dans `$tab2` (peu importe le type d'indice) :

```
$tab1 = array ("rouge", "jaune", "aa"=>"bleu");
$tab2 = array ("bb"=>"jaune","bleu");
$tab = array_diff($tab1,$tab2);
// $tab contient :  Array ( [0] => rouge )
```

array_fill_keys()

- Signature : `array_fill_keys($tab_clefs, $valeur)`.
- Versions : PHP 5 >= 5.2.0.

Cette fonction retourne un tableau dont les clefs correspondent aux valeurs du tableau `$tab_clefs` et dont les valeurs correspondent à `$valeur` :

```
$tab_clefs = array("a", "b", "c");
$tab = array_fill_keys($tab_clefs, 1);
// $tab contient : Array ( [a] => 1 [b] => 1 [c] => 1 )
```

array_filter()

- Signature : `array_filter($tab, $fct)`.
- Versions : PHP 4 à partir de la 4.0.6, PHP 5.

Cette fonction retourne un tableau qui ne contient que les éléments `$tab` qui ont été validés par la fonction `$fct` :

```
function primaire($val)
{
  if (($val == "rouge") || ($val == "jaune") || ($val ==
  ✂ "bleu"))
    return (1);
}

$tab = array ("rouge","orange","marron","bleu");
$tab2 = array_filter($tab,"primaire");
// seules les couleurs primaires sont conservées
// $tab2 contient :
// Array ( [0] => rouge [3] => bleu )


function pair($val)
{
  if (($val % 2))
    return (1);
}

$tab = array (1,2,3,4,5,6,7);
$tab2 = array_filter($tab,"pair");
// seuls les nombres impairs sont conservés
// $tab2 contient :
// Array ( [0] => 1 [2] => 3 [4] => 5 [6] => 7 )
```

Une fonction interne à PHP peut également être utilisée en tant que *callback* :

```
$tab = array ("bonjour",3,3.2,0,false);
$tab2 = array_filter($tab,"is_int");
// seuls les entiers sont conservés
// $tab2 contient :
// Array ( [1] => 3 [3] => 0 )
```

array_flip()

- Signature : `array_flip ($tab).`
- Versions : PHP 4 à partir de la 4.0b4, PHP 5.

Cette fonction retourne un tableau dont les indices sont devenus les valeurs, et inversement :

```
$tab = array ("rouge","aa"=>"orange","marron","bleu");
$tab2 = array_flip($tab);
print $tab["aa"];  // affiche : "orange"
print $tab2["orange"];  // affiche : "aa"
```

array_intersect()

- Signature : `array_intersect ($tab1,$tab2).`
- Versions : PHP 4 à partir de la 4.0.1, PHP 5.

Cette fonction retourne un tableau contenant les valeurs présentes à la fois dans `$tab1` et dans `$tab2` :

```
$tab1 = array ("rouge", "jaune", "aa"=>"bleu");
$tab2 = array ("bb"=>"jaune","bleu");
$tab = array_intersect($tab1,$tab2);
// $tab contient :  Array ( [1] => jaune [aa] => bleu )
```

array_keys()

- Signature : `array_keys ($tab).`
- Versions : PHP 4, PHP 5.

Cette fonction retourne un tableau contenant les indices de `$tab` :

```
$tab = array ("rouge","aa"=>"orange","marron","bleu");
$tab2 = array_keys($tab);
// $tab2 contient les valeurs 0, "aa", 1, 2
```

Il est possible de préciser en paramètre la valeur dont on veut récupérer les indices :

```
$tab = array ("rouge","aa"=>"orange","marron","bleu",
✂ "orange");
$tab2 = array_keys($tab,"orange");
// $tab2 contient les valeurs "aa" et 3
```

array_map()

- Signature : `array_map ($fct,$tab [, $tab2])`.
- Versions : PHP 4 à partir de la 4.0.6, PHP 5.

Cette fonction permet d'appliquer une fonction `$fct` à tous les éléments d'un tableau `$tab` :

```
function double($val)
{
  return $val * $val;
}

$tab = array ( 1, 2, 3);
$tab2 = array_map("double",$tab);
// $tab2 contient : 1, 4, 9
```

Il est aussi possible de transmettre plusieurs tableaux à `array_map()`. La fonction *callback* `$fct` doit alors avoir autant de paramètres qu'il y a de tableaux :

```
function sup ($x, $y)
{
  if ($x > $y) return $x;
  return $y;
}

$tab1 = array (2,18,-10,2,6);
$tab2 = array (12, 2, 4, 34, -3);
$max = array_map("sup",$tab1,$tab2);
// le tableau $max contient les plus grandes valeurs de
// $tab1 et de $tab2 :
// Array ( [0] => 12 [1] => 18 [2] => 4 [3] => 34 [4] => 6 )
```

array_merge()

- Signature : `array_merge ($tab1, $tab2…)`.
- Versions : PHP 4, PHP 5.

Cette fonction retourne un tableau composé des éléments des différents tableaux passés en paramètres :

```
$tab1 = array ("rouge", "jaune", "bleu");
$tab2 = array ("vert", "blanc");
$tab = array_merge($tab1,$tab2);
// $tab contient :
// Array ( [0] => rouge [1] => jaune [2] => bleu [3]
//          => vert [4] => blanc )
```

array_merge_recursive()

- Signature : `array_merge_recursive ($tab1, $tab2…)`.
- Versions : PHP 4 à partir de la 4.0.1, PHP 5.

Comme `array_merge()`, cette fonction fusionne des tableaux. Cependant, si l'un des tableaux contient d'autres tableaux, les éléments de ces derniers seront pris en compte dans la fusion :

```
$tab1 = array ("rouge", "jaune", "aa"=>array("bleu",
✂ "violet"));
$tab2 = array ("vert", "aa"=>array("cyan"), "blanc");
$tab = array_merge_recursive($tab1,$tab2);
// $tab contient :
// Array ( [0] => rouge [1] => jaune [aa] =>
//   Array ( [0] => bleu [1] => violet [2] => cyan )
//   [2] => vert [3] => blanc )
```

array_pad()

- Signature : `array_pad ($tab, $tailler, $val)`.
- Versions : PHP 4 à partir de la 4.0b4, PHP 5.

Cette fonction permet d'augmenter la taille d'un tableau en comblant les espaces avec la valeur `$val`. Si la variable `$taille` est négative, l'insertion des nouvelles valeurs se fait au début :

```
$tab = array (1, 2 ,3);
$tab2 = array_pad($tab,5,9);
// $tab2 contient : 1, 2, 3, 9, 9
$tab3 = array_pad($tab,-5,9);
// $tab3 contient : 9, 9, 1, 2, 3
```

array_pop()

- Signature : `array_pop ($tab)`.
- Versions : PHP 4, PHP 5.

Cette fonction retourne la dernière valeur du tableau `$tab`. Le paramètre `$tab` est alors écourté de cette dernière valeur :

```
$tab = array ("rouge","jaune","bleu");
$dernier = array_pop($tab);
// la variable $dernier vaut "bleu"
// et $tab ne contient plus que "rouge", "jaune"
```

array_push()

- Signature : `array_push ($tab, $val1, $val2…)`.
- Versions : PHP 4, PHP 5.

Cette fonction ajoute des éléments à un tableau :

```
$tab = array ("rouge", "jaune");
array_push($tab,"bleu","vert");
// $tab contient rouge, jaune, bleu, vert
```

Il aurait également été possible d'ajouter les deux derniers éléments de la manière suivante :

```
$tab = array ("rouge", "jaune");
$tab[] = "bleu";
$tab[] = "vert";
```

array_rand ()

- Signature : `array_rand ($tab)`.
- Versions : PHP 4, PHP 5.

Cette fonction retourne un indice aléatoire du tableau `$tab`.

La fonction `srand()` doit être appelée préalablement :

```
srand ((double) microtime() * 10000000);
$tab = array ("rouge","jaune","bleu");
$indice = array_rand($tab);
print $tab[$indice];
// affiche une des valeurs de $tab
```

array_reverse()

- Signature : `array_reverse ($tab).`
- Versions : PHP 4 à partir de la 4.0b4, PHP 5.

Cette fonction retourne un tableau contenant les éléments de `$tab` dans l'ordre inverse :

```
$tab = array ("rouge","jaune","bleu");
$tab2 = array_reverse($tab);
// $tab2 contient bleu, jaune, rouge
```

array_reduce()

- Signature : `array_reduce ($tab,$func).`
- Versions : PHP 4 à partir de la 4.0.5, PHP 5.

Cette fonction permet de réduire le tableau à une seule valeur en appliquant la fonction `$func` itérativement à tous les éléments du tableau `$tab` :

```
function mult($a,$b)
{
  $a *= $b;
  return $a;
}
$tab = array(2,4,6);
echo array_reduce($tab,$mult);
// affiche 48 : 2 * 4 * 6
```

array_shift()

- Signature : `array_shift ($tab).`
- Versions : PHP 4, PHP 5.

Cette fonction permet d'extraire le premier élément du tableau `$tab` et de le retourner :

```
$tab = array("www","kernix","com");
echo array_shift($tab);
// affiche "www", $tab ne contient plus que "kernix" et
✂ "com"
```

array_slice()

- Signature : `array_slice ($tab,$deb[,$n])`.
- Versions : PHP 4, PHP 5.

Cette fonction permet d'extraire un sous-tableau du tableau `$tab`.

Si `$deb` est positif, `$tab2` contient les éléments commençant à l'indice `$deb` du tableau `$tab` ; si `$deb` est négatif, l'indice est à considérer à partir de la fin de `$tab`.

Si `$n` est positif, le sous-tableau contiendra `$n` éléments.

Si `$n` est négatif, les `$n` derniers éléments de `$tab` ne feront pas partie du sous-tableau.

Si `$n` n'est pas précisé, tous les éléments, à partir de l'indice `$deb`, seront retournés.

```
$tab = array(1,2,3,4,5,6,7,8,9);
$tab2 = array_slice($tab,2);  // $tab2 contient
✂ 3,4,5,6,7,8,9
$tab2 = array_slice($tab,2,2);  // $tab2 contient 3,4
$tab2 = array_slice($tab,2,-2);  // $tab2 contient
✂ 3,4,5,6,7
$tab2 = array_slice($tab,-2,2);  // $tab2 contient 8,9
```

array_splice()

- Signature : `array_splice ($tab, $offset [, $ln, $remplacement])`.
- Versions : PHP 4, PHP 5.

Cette fonction supprime une partie d'un tableau et la remplace par un autre élément.

Si `$offset` est positif, la portion à supprimer commence à l'index `$offset`. Si `$offset` est négatif l'index est à prendre à partir de la fin du tableau :

```
$tab = array ("a","b","c","d","e");
$r = array_splice($tab, 3);
// $tab contient : Array ( [0] => a [1] => b [2] => c )
// $r contient : Array ( [0] => d [1] => e )
```

```
$tab = array ("a","b","c","d","e");
$r = array_splice($tab, -3);
// $tab contient : Array ( [0] => a [1] => b )
// $r contient : Array ( [0] => c [1] => d [2] => e )
```

Si le paramètre `$ln` est précisé et est positif, seuls `$ln` éléments de `$tab` seront supprimés. S'il est négatif, la portion supprimée s'arrête `$ln` éléments avant la fin de `$tab` :

```
$tab = array ("a", "b", "c", "d", "e", "f");
$r = array_splice($tab, 2, 1);
// $tab contient : Array ( [0] => a [1] => b [2] => d
//                         [3] => e [4] => f )
// $r contient : Array ( [0] => c )
$tab = array ("a", "b", "c", "d", "e", "f");
$r = array_splice($tab, 2, -1);
// $tab contient : Array ( [0] => a [1] => b [2] => f )
// $r contient : Array ( [0] => c [1] => d [2] => e )
```

Si le paramètre `$remplacement` est précisé, les éléments retirés de `$tab` sont remplacés par les éléments de `$remplacement` :

```
$tab = array ("a", "b", "c", "d", "e", "f");
$remplacement = array (1, 2);
$r = array_splice($tab, 2, 1, $remplacement);
// $tab contient : Array ( [0] => a [1] => b [2] => 1 [3] => 2
                           [4] => d [5] => e [6] => f )
// $r contient : Array ( [0] => c )
```

Si juste un élément de `$tab` doit être remplacé par un seul élément, `$remplacement` peut être une valeur simple.

Si les valeurs de `$offset` et de `$ln` font qu'aucun élément ne doit être remplacé dans `$tab`, `$remplacement` est alors inséré à la position `$offset` :

```
$tab = array ("a", "b", "c", "d", "e", "f");
$remplacement = array (1, 2);
$r = array_splice($tab, 2, -50, $remplacement);
// $tab contient : Array ( [0] => a [1] => b [2] => 1 [3] => 2
                       [4] => c [5] => d [6] => e [7] => f )
// $r est vide
```

array_sum()

- Signature : `array_sum ($tab)`.
- Versions : PHP 4 à partir de la 4.0.4, PHP 5.

Cette fonction additionne les valeurs de `$tab` :

```
$tab = array ("1", 2, 3);
print array_sum ($tab); // affiche 6
```

array_unique()

- Signature : `array_unique ($tab)`.
- Versions : PHP 4, PHP 5.

Cette fonction supprime les doublons d'un tableau :

```
$tab = array ("1", 2, "coucou", 1, 'coucou', 1.0);
print_r(array_unique ($tab));
// affiche : Array ( [0] => 1 [1] => 2 [2] => coucou )
```

array_unshift()

- Signature : `array_unshift ($tab, $elem1 [, $elem2…])`.
- Versions : PHP 4, PHP 5.

Cette fonction permet d'ajouter des éléments au début d'un tableau :

```
$tab = array ("a","b");
array_unshift ($tab, "c", array ("d", "e"));
/* $tab contient
Array
(
    [0] => c
    [1] => Array
        (
            [0] => d
            [1] => e
        )
    [2] => a
    [3] => b
)
*/
```

La fonction retourne le nouveau nombre d'éléments dans le tableau.

array_values()

- Signature : `array_values ($tab)`.
- Versions : PHP 4, PHP 5.

Cette fonction retourne les valeurs contenues dans `$tab` :

```
$tab = array ("a", "b"=>"c", 1);
print_r(array_values($tab));
/* affiche :
Array
(
    [0] => a
    [1] => c
    [2] => 1
)
*/
```

array_walk()

- Signature : `array_walk ($tab, $fct)`.
- Versions : PHP 3 à partir de la 3.0.3, PHP 4, PHP 5.

Cette fonction permet d'appliquer la fonction `$fct` à tous les éléments de `$tab`. La fonction `$fct` reçoit en premier paramètre la valeur, et en deuxième la clé (l'index) de l'élément :

```
function nbcar ($valeur, $clef)
{
  echo strlen($valeur), "<br>";
}

$tab = array ("BonJour", "BONJ");
array_walk($tab,"nbcar");
/* affiche :
7
4
*/
```

Pour modifier directement les valeurs de `$tab`, il est nécessaire que le premier paramètre de `$fct` soit passé par référence :

```
function minuscule (&$valeur, $clef)
{
  $valeur = strtolower($valeur);
}

$tab = array ("BonJour", "BONJOUR");
array_walk($tab,"minuscule");
/* $tab contient :
Array
(
    [0] => bonjour
    [1] => bonjour
```

```
)
*/
```

asort()

- Signature : `asort ($tab).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction trie les éléments d'un tableau (en conservant l'association indice/valeur) :

```
$tab = array ("a" => "hello", "bonjour", "b" => "ola");
asort($tab);
/* $tab contient :
Array
(
    [0] => bonjour
    [a] => hello
    [b] => ola
)
*/
```

La fonction `arsort()` trie, quant à elle, les éléments en ordre inverse :

```
$tab = array ("a" => "hello", "bonjour", "b" => "ola");
arsort($tab);
/* $tab contient :
Array
(
    [b] => ola
    [a] => hello
    [0] => bonjour
)
*/
```

compact()

- Signature : `compact ($var1 [, $var2…]).`
- Versions : PHP 4, PHP 5.

Cette fonction permet de créer un tableau associatif à partir des noms de variables :

```
$prenom = "Paul";
$nom = "Dupont";
$tab = compact("nom","prenom");
/* $tab contient :
```

```
Array
(
    [nom] => Dupont
    [prenom] => Paul
)
*/
```

count()

- Signature : `count ($tab)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne le nombre d'éléments d'un tableau :

```
$tab = array ("a", "b", "c");
echo count($tab); // affiche 3
$tab[1] = "a";
$tab[5] = "b";
$tab[6] = "c";
echo count($tab); // affiche 5
```

La fonction `sizeof()` est un alias de `count()`.

current()

- Signature : `current ($tab)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne l'élément « courant » du tableau `$tab`. Cet élément est celui qui est associé au pointeur interne du tableau.

Ce pointeur peut être modifié par les fonctions suivantes :

- `end()` place le pointeur interne à la fin du tableau.
- `next()` avance d'une position le pointeur interne du tableau.
- `prev()` recule d'une position le pointeur interne du tableau.
- `reset()` place le pointeur interne au début du tableau.

Le script suivant permet d'afficher tous les éléments d'un tableau :

```
$tab = array ("a", "b", "c", "d");
while ($val = current($tab))
{
  echo "$val - ";
```

```
  next ($tab);
}
// affiche : a - b - c - d -
```

Ce script affiche, quant à lui, les valeurs de `$tab` dans l'ordre inverse :

```
$tab = array ("a", "b", "c", "d");
end ($tab);
while ($val = current($tab))
{
  echo "$val - ";
  prev ($tab);
}
// affiche : d - c - b - a -
```



Élément vide

Si un des éléments du tableau contient les valeurs `0` ou `""`, la fonction `current()` retourne alors `false`.

La fonction `pos()` est un alias de `current()`.

each()

- Signature : `each ($tab)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne un tableau contenant l'index et la valeur courante du tableau `$tab`, puis avance le pointeur d'une position :

```
$tab = array ("prenom" => "Claire", "nom" => "Fulchiron");
next ($tab);
print_r(each ($tab));
/* affiche
Array
(
    [1] => Fulchiron
    [value] => Fulchiron
    [0] => nom
    [key] => nom
)
*/
```

Cette fonction permet de traverser tout un tableau sans se soucier des valeurs contenues dans ce dernier :

```
$tab = array ("prenom" => "Paul", "nom" => "Dupont", 0,
✂ "annee" => 1977, "Paris");
while (list ($clef, $valeur) = each ($tab))
{
   echo "$clef > $valeur<br>";
}
/* affiche
prenom > Paul
nom > Dupont
0 > 0
annee > 1977
1 > Paris
*/
```

extract()

- Signature : `extract ($tab [, $mode [, $prefixe]])`.
- Versions : PHP 3 à partir de la 3.0.7, PHP 4, PHP 5.

Cette fonction permet de créer et d'initialiser des variables à partir des clés et des valeurs des éléments d'un tableau associatif.

Quatre modes sont disponibles…

- `EXTR_OVERWRITE` : si une variable du même nom existe, son contenu est remplacé (mode par défaut).
- `EXTR_SKIP` : si une variable de même nom existe, elle n'est pas modifiée.
- `EXTR_PREFIX_SAME` : si une variable de même nom existe, une nouvelle variable est créée (en utilisant le préfixe `$prefixe`).
- `EXTR_PREFIX_ALL` : préfixe toutes nouvelles variables avec `$prefixe`.

La fonction `extract()` retourne le nombre de variables importées avec succès :

```
$nom = "Durand";
$tab = array ("prenom" => "Paul", "nom" => "Dupont");
extract($tab);
echo $nom;  // affiche : Dupont
```

```
$nom = "Durand";
$tab = array ("prenom" => "Paul", "nom" => "Dupont");
extract($tab,EXTR_SKIP);
echo $nom;  // affiche : Durand
```

```
$nom = "Durand";
$tab = array ("prenom" => "Paul", "nom" => "Dupont");
extract($tab, EXTR_PREFIX_SAME, "var_");
echo $nom;  // affiche : Durand
echo $var_nom;  // affiche : Dupont
$nom = "Durand";
$tab = array ("prenom" => "Paul", "nom" => "Dupont");
extract($tab, EXTR_PREFIX_SAME, "var");
echo $nom;  // affiche : Durand
echo $var_nom;  // affiche : Dupont
$nom = "Durand";
$tab = array ("prenom" => "Paul", "nom" => "Dupont");
extract($tab, EXTR_PREFIX_ALL, "var");
echo $nom;  // affiche : Durand
echo $var_nom;  // affiche : Dupont
echo $var_prenom;  // affiche : Paul
```

in_array()

- Signature : `in_array ($var, $tab [, $strict])`.
- Versions : PHP 4, PHP 5.

Cette fonction retourne `true` si la variable `$var` est présente dans le tableau `$tab` :

```
$tab = array ("prenom" => "Paul", "nom" => "Dupont",
✂ "annee" => 1977);
if (in_array("1977", $tab)) echo "élément trouvé";
else echo "élément absent";
// affiche : "élément trouvé"
```

Si le paramètre `$strict` (booléen) est précisé, les types doivent aussi correspondre :

```
$tab = array ("prenom" => "Paul", "nom" => "Dupont",
✂ "annee" => 1977);
if (in_array("1977", $tab, true)) echo "élément trouvé";
else echo "élément absent";
// affiche : "élément absent" car vous avez un numérique
✂ et une chaîne
```

array_search()

- Signature : `array_search ($var, $tab [, $strict])`.
- Versions : PHP 4 à partir de la 4.0.5, PHP 5.

Cette fonction recherche une valeur (`$var`) dans un tableau et retourne, en cas de succès, l'indice de l'élément trouvé.

Si le paramètre booléen `$strict` est précisé, une comparaison de type est effectuée :

```
$tab = array ("prenom" => "Paul", "nom" => "Dupont",
✂ "annee" => 1977);
echo array_search ("1977",$tab);  // affiche : annee
```

key()

- Signature : `key ($tab)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne l'indice de la position courante :

```
$tab = array ("prenom" => "Paul", "nom" => "Dupont",
✂ "annee" => 1977);
next($tab);
echo key($tab);  // affiche : nom
```

ksort()

- Signature : `ksort ($tab)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de classer un tableau par clés (la corrélation clé/valeur est maintenue) :

```
$tab = array ("prenom" => "Paul", "nom" => "Dupont",
✂ "annee" => 1977);
ksort($tab);
/* $tab contient :
Array
(
    [annee] => 1977
    [nom] => Dupont
    [prenom] => Paul
)
*/
```

La fonction `krsort()` classe en ordre inverse :

```
$tab = array ("prenom" => "Paul", "nom" => "Dupont",
✂ "annee" => 1977);
```

```
krsort($tab);
/* $tab contient :
Array
(
    [prenom] => Paul
    [nom] => Dupont
    [annee] => 1977
)
*/
```

list()

- Signature : `list ()`.

Cette fonction permet, en une ligne, d'assigner des valeurs à plusieurs variables :

```
$tab = array ("prenom" => "Paul", "nom" => "Dupont",
✂ "annee" => 1977);
list ($prenom, $nom) = array_values ($tab);
echo $nom;
```

natsort()

- Signature : `natsort ($tab)`.
- Versions : PHP 4, PHP 5.

Cette fonction permet de classer les éléments d'un tableau en utilisant l'ordre naturel :

```
$tab = array ("img21.png","img12.png","img1.png","img2.png");
natsort($tab);
/* $tab contient :
Array
(
    [2] => img1.png
    [3] => img2.png
    [1] => img12.png
    [0] => img21.png
)
*/
```

```
$tab = array ("img21.png","img12.png","img1.png",
✂ "img2.png");
sort($tab);
/* $tab contient :
Array
(
    [0] => img1.png
    [1] => img12.png
    [2] => img2.png
    [3] => img21.png
)
*/
```

`natcasesort()` fonctionne sur le même modèle que `natsort()`, sans être sensible à la casse.

range()

- Signature : `range ($min, $max).`
- Versions : PHP 3 à partir de la 3.0.8, PHP 4, PHP 5.

Cette fonction permet de remplir un tableau à partir d'un intervalle :

```
$tab = range (12, 16);
echo $tab[1];  // affiche 13
```

Avec une version de PHP supérieure ou égale à 4.1.0, il est possible d'écrire `range ('a', 'z').`

shuffle()

- Signature : `shuffle ($tab).`
- Versions : PHP 3 à partir de la 3.0.8, PHP 4, PHP 5.

Cette fonction permet de mélanger les éléments d'un tableau.

Le générateur de nombres aléatoires doit être initialisé auparavant :

```
$tab = range (1, 10);
srand ((float)microtime()*1000000);
shuffle ($tab);
while (list ($clef, $valeur) = each ($tab)) print("$valeur
✂ - ");
// affiche : 4 - 6 - 5 - 10 - 8 - 2 - 3 - 1 - 9 - 7 -
```

sort()

- Signature : `sort ($tab [, $mode]).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de classer les éléments d'un tableau.

La fonction `rsort()` classe, quant à elle, en ordre inverse.

Le paramètre optionnel `$mode` (ajouté aux versions de PHP supérieures ou égales à 4) permet de spécifier le type de comparaison qui est utilisée pour le tri.

- `SORT_REGULAR` : comparaison normale.
- `SORT_NUMERIC` : comparaison numérique.
- `SORT_STRING` : comparaison de chaînes.

```
$tab = array (2, "1", 3.0, -12, "12", "-5");
sort($tab, SORT_STRING);
/* $tab contient
Array
(
    [0] => -12
    [1] => -5
    [2] => 1
    [3] => 12
    [4] => 2
    [5] => 3
)
*/

$tab = array (2, "1", 3.0, -12, "12", "-5");
sort($tab, SORT_NUMERIC);
/* $tab contient
Array
(
    [0] => -12
    [1] => -5
    [2] => 1
    [3] => 2
    [4] => 3
    [5] => 12
)
*/
```

uasort()

- Signature : `uasort ($tab, $fct).`
- Versions : PHP 3 à partir de la 3.0.4, PHP 4, PHP 5.

Cette fonction permet de trier un tableau en précisant la fonction de comparaison `$fct`.

Dans l'exemple suivant, vous classez les éléments du tableau par longueur de chaîne :

```
function comp_long($x, $y)
{
  if (strlen($x) > strlen($y)) return 1;
  return -1;
}

$tab = ("z" => "11", 9,  "b" => "aaaa", "q" => "zzz");
usort($tab, "comp_long");

/* $tab contient :
Array
(
    [0] => 9
    [1] => 11
    [2] => zzz
    [3] => aaaa
)
*/
```

La fonction `uasort()` maintient l'association clé/valeur :

```
function comp_long($x, $y)
{
  if (strlen($x) > strlen($y)) return 1;
  return -1;
}

$tab = ("z" => "11", 9,  "b" => "aaaa", "q" => "zzz");
usort($tab, "comp_long");

/* $tab contient :
Array
(
    [0] => 9
    [z] => 11
    [q] => zzz
    [b] => aaaa
)
*/
```

La fonction `uksort()` trie, quant à elle, les clés.

18.5. Les fonctions de dates et d'heures

Il est courant en informatique de faire référence à un *timestamp*. Il s'agit d'un nombre entier qui correspond au nombre de secondes entre le 1er janvier 1970 et une date donnée.

checkdate()

- Signature : `checkdate ($mois, $jour, $annee)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne `true` si la date est valide, sinon `false`.

- L'année doit être comprise entre 1 et 32767.
- Le mois doit être compris entre 1 et 12.
- Le jour doit être une valeur autorisée pour le mois donné (les années bissextiles sont prises en compte).

date()

- Signature : `date ($format [, $timestamp])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet d'afficher une date (*timestamp*) suivant un certain format.

Si le paramètre `$timestamp` n'est pas précisé, c'est la date courante qui est prise en compte.

-

Les caractères de substitution utilisés au sein du format sont listés dans le chapitre consacré aux Dates et Heures

```
print (date("s"));  // affiche le nombre de secondes de la
✂ date actuelle
echo date("d/m/Y");  // affiche la date à la française,
✂ par exemple 12/03/2001
```

Il est courant d'utiliser la fonction `date()` avec la fonction `mktime()` :

```
$demain  = mktime (0, 0, 0, date("m"), date("d") + 1,
✂ date("Y"));
$moisdernier = mktime (0, 0, 0, date("m")-1, date("d"),
✂ date("Y"));
$anneeprochaine  = mktime(0, 0, 0, date("m"), date("d"),
✂ date("Y") + 1);
$demain = mktime (0, 0, 0, date("m"), date("d") + 1,
✂ date("Y"));
echo "demain nous serons le " . date("j",$demain);
// si nous sommes le 31, affiche : "demain nous serons le 1"
```

Pour travailler avec un temps GMT (Greenwich Mean Time), la fonction `gmdate()` doit être utilisée.

getdate()

- Signature : `getdate ([$timestamp]).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne un tableau associatif contenant des informations sur la date courante (ou sur le *timestamp* si le paramètre est précisé).

- `"seconds"` : secondes.
- `"minutes"` : minutes.
- `"hours"` : heures.
- `"mday"` : jour du mois.
- `"wday"` : jour de la semaine, numérique, de `0` (dimanche) à `6` (samedi).
- `"mon"` : mois, numérique.
- `"year"` : année, numérique.
- `"yday"` : jour de l'année, numérique, c'est-à-dire `"299"`.
- `"weekday"` : jour de la semaine, texte complet (en anglais), c'est-à-dire `"Friday"`.
- `"month"` : mois, texte complet, en anglais, c'est-à-dire `"January"`.

```
$aujourdhui = getdate();
$mois = $aujourdhui['month'];
$mjour = $aujourdhui['mday'];
$annee = $aujourdhui['year'];
echo "$mjour/$mois/$annee";  // affiche la date à la
✂ française
```

microtime()

- Signature : `microtime ().`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne le *timestamp* de la date courante avec les microsecondes. La fonction retourne, en fait, une chaîne de caractères formatée de la façon suivante : `"msec sec"` où `"sec"` correspond au *timestamp* en secondes de la date courante et `"msec"` correspond aux millisecondes.

mktime()

- Signature : `mktime ($heure, $minute, $seconde, $mois, $jour, $annee).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de retourner le *timestamp* d'une date :

```
echo date ("l", mktime (0,0,0,1,1,2000));  // affiche
✂ "Saturday" et permet ainsi de savoir que le 1er janvier
✂ 2000 était un samedi
```

Les paramètres transmis à `mktime()` ne sont pas obligatoirement « valides » (le mois n'est pas obligatoirement compris entre 1 et 12). Tous les exemples suivants affichent `"01/Jan/1998"` :

```
echo date ("d/M/Y", mktime (0,0,0,12,32,1997));
echo date ("d/M/Y", mktime (0,0,0,13,1,1997));
echo date ("d/M/Y", mktime (0,0,0,1,1,1998));
echo date ("d/M/Y", mktime (0,0,0,1,1,98));
```

La fonction `gmmktime()` permet de travailler avec des dates GMT.

strftime()

- Signature : `strftime ($format [, $timestamp]).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de formater une date selon la langue locale.

Les caractères de conversion qui peuvent être contenus dans le format sont…

- %a : nom abrégé du jour de la semaine (local).
- %A : nom complet du jour de la semaine (local).
- %b : nom abrégé du mois (local).
- %B : nom complet du mois (local).
- %c : représentation préférée pour les dates et les heures, en local.
- %C : numéro de siècle (l'année, divisée par 100 et arrondie entre 00 et 99).
- %d : jour du mois en numérique (intervalle de 01 à 31).
- %D : identique à %m/%d/%y.
- %e : numéro du jour du mois, les chiffres sont précédés d'un espace (de '1' à '31').
- %h : identique à %b.
- %H : heure de la journée en numérique et sur 24 heures (intervalle de 00 à 23).
- %I : heure de la journée en numérique et sur 12 heures (intervalle de 01 à 12).
- %j : jour de l'année, en numérique (intervalle de 001 à 366).
- %m : mois en numérique (intervalle de 1 à 12).
- %M : minute en numérique.
- %n : *newline character*.
- %p : soit 'am' ou 'pm', en fonction de l'heure absolue ou en fonction des valeurs enregistrées en local.
- %r : l'heure au format AM et PM.
- %R : l'heure au format 24 h.
- %S : secondes en numérique.
- %t : tabulation.
- %T : l'heure actuelle (égale à %H:%M:%S).
- %u : le numéro du jour dans la semaine de 1 à 7 (1 représente lundi).
- %U : numéro de semaine dans l'année, en considérant le premier dimanche de l'année comme le premier jour de la première semaine.
- %V : le numéro de semaine comme défini dans la norme ISO 8601 (1988), sous forme décimale, de 01 à 53. La semaine 1 est la

première semaine qui a plus de quatre jours dans l'année courante et dont lundi est le premier jour.

- %W : numéro de semaine dans l'année, en considérant le premier lundi de l'année comme le premier jour de la première semaine.
- %w : jour de la semaine, numérique (0 représente dimanche).
- %x : format préféré de représentation de la date sans l'heure.
- %X : format préféré de représentation de l'heure sans la date.
- %y : l'année, numérique, sur deux chiffres (de 00 à 99).
- %Y : l'année, numérique, sur quatre chiffres.
- %Z : fuseau horaire, ou nom ou abréviation.
- %% : un caractère '%' littéral.

L'exemple suivant permet d'écrire le jour courant dans trois langues différentes :

```
print (strftime ("Le jour %A se dit en :<br><br>"));

setlocale ("LC_TIME", "fi_FI");
print (strftime ("- finlandais : %A<br>"));

setlocale ("LC_TIME", "fr_CA");
print (strftime ("- français : %A<br>"));

setlocale ("LC_TIME", "de_DE");
print (strftime ("- allemand : %A<br>"));

/* affiche par exemple :
Le jour Sunday se dit en :
- finlandais : sunnuntai
- français : dimanche
- allemand : Sonntag
*/
```

La fonction gmstrftime() permet de travailler avec des dates GMT.

time()

- Signature : time ().
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne le *timestamp* de la date courante.

strtotime()

- Signature : `strtotime ($str)`.
- Versions : PHP 3 à partir de la 3.0.12, PHP 4, PHP 5.

Cette fonction essaie de convertir une date exprimée en anglais usuel :

```
echo date ("l", strtotime ("1 January 3000"));
// le 1er janvier 3000 sera un jeudi
```

18.6. Les fichiers et les répertoires

PHP permet de manipuler les fichiers, qu'ils soient locaux ou distants (sur d'autres serveurs).

Dans cette partie, le paramètre `$fichier` correspond à un nom de fichier (`"toto.txt"`, `"/tmp/toto.txt"`, `"http://www.site.com/toto.txt"`) et `$pfichier` correspond, lui, à un pointeur sur fichier (un identifiant de fichier).

Il en va de même pour les paramètres `$repertoire` et `$prep`.

basename()

- Signature : `basename ($chemin [, $suffixe])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne le nom du fichier contenu dans le chemin `$chemin`. Si le paramètre `$suffixe` est précisé, le suffixe du nom de fichier n'est pas indiqué.

```
$chemin = "/sites/monsite/images/titre.gif";
echo basename ($chemin);  // affiche : titre.gif
echo basename ($chemin, ".gif");  // affiche : titre
```

Attention, un répertoire peut être considéré comme un fichier, il est donc possible d'écrire :

```
$chemin = "/sites/monsite/images";
echo basename ($chemin);  // affiche : images
```

Les barres obliques inversées ou non peuvent être utilisées sous Windows comme caractères de séparation dans le chemin d'accès. Seule

la barre oblique simple est acceptée sur les autres systèmes d'exploitation.

chdir()

- Signature : `chdir ($repertoire).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de changer de répertoire courant. Elle retourne `false` en cas d'erreur, sinon `true`.

chgrp()

- Signature : `chgrp ($fichier, $groupe).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de changer le groupe d'un fichier. Si vous n'êtes pas super-utilisateur, vous devez à la fois être membre du groupe de départ (celui du fichier) et d'arrivée (`$groupe`) pour pouvoir le modifier.

La fonction retourne `true` en cas de succès, sinon `false`. La variable `$groupe` peut contenir l'`id` du groupe. Cette fonction ne fonctionne pas sous Windows.

chmod()

- Signature : `chmod ($fichier, $droits).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de changer les droits d'un fichier :

```
chmod ("/sites/monsite/images/titre.gif", 0644);
/* les droits du fichier passent en :
- lecture, écriture pour le propriétaire du fichier [6 : 110]
- lecture pour le groupe du propriétaire [4 : 100]
- lecture pour tout le monde [4 : 100]
*/
```

La fonction retourne `true` en cas de succès, sinon `false`. Elle ne fonctionne pas sous Windows.

chown()

- Signature : `chown ($fichier, $utilisateur).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de changer le propriétaire du fichier. Seul le super-utilisateur peut modifier le propriétaire d'un fichier.

La fonction retourne `true` en cas de succès, sinon `false`. Elle ne fonctionne pas sous Windows.

clearstatcache()

- Signature : `clearstatcache ().`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de vider le cache système qui a été mis en place lors de l'accès à un fichier. Cette fonction est particulièrement utile lorsque vous travaillez sur des fichiers susceptibles de changer très souvent.

Les fonctions qui peuvent avoir besoin d'être suivies de `clearstatcache()` sont : `stat()`, `lstat()`, `file_exists()`, `is_writable()`, `is_readable()`, `is_executable()`, `is_file()`, `is_dir()`, `is_link()`, `filectime()`, `fileatime()`, `filemtime()`, `fileinode()`, `filegroup()`, `fileowner()`, `filesize()`, `filetype()` et `fileperms()`.

closedir()

- Signature : `closedir ($prep).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de fermer l'identifiant de répertoire `$prep`.

copy()

- Signature : `copy ($fichierorig, $fichierdest).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de copier un fichier d'un emplacement de départ vers un emplacement d'arrivée.

La fonction retourne `true` en cas de succès, sinon `false`.

```
if (!copy("/sites/monsite/images/titre.gif","/sites/monsite
✂ /titre.gif"))
{
  print("la copie a échoué");
}
else
{
  print("le fichier a été déplacé dans le répertoire
  ✂ /sites/monsite");
}
```

delete()



Reportez-vous à la fonction `unlink()`.

dirname()

- Signature : `dirname ($chemin)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne le répertoire contenu dans `$chemin` :

```
$chemin = "/sites/monsite/images";
echo dirname ($chemin);  // affiche : /sites/monsite
```

unlink()

- Signature : `unlink ($fichier)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet d'effacer un fichier.

La fonction `delete()` est un alias de la fonction `unlink()`.

Cette fonction retourne `0` ou `false` en cas d'erreur. Elle n'est pas prise en charge sous Windows.

disk_free_space()

- Signature : `disk_free_space($repertoire)`.
- Versions : PHP 4 à partir de la 4.0.7RC1, PHP 5.

Cette fonction retourne la place disponible sur la partition où se trouve le répertoire (en octets).

La fonction `diskfreespace()` est un alias de `disk_free_space()`.

disk_total_space()

- Signature : `disk_total_space($repertoire)`.
- Versions : PHP 4 à partir de la 4.0.7RC1, PHP 5.

Cette fonction retourne la taille totale de la partition où se trouve le répertoire.

fclose()

- Signature : `fclose ($pfichier)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de fermer un pointeur sur fichier (un identifiant de fichier). Elle retourne `true` en cas de succès, sinon `false`.

feof()

- Signature : `feof ($pfichier)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne `true` si le pointeur du fichier `$pfichier` est à la fin du fichier (le caractère `EOF`, ou *end of file*, est rencontré), sinon `false`.

fflush()

- Signature : `fflush ($pfichier).`
- Versions : PHP 4, PHP 5.

Cette fonction force l'écriture de toutes les données « bufférisées » dans le fichier pointé par `$pfichier`. Elle retourne `true` en cas de succès, sinon `false`.

fgetc()

- Signature : `fgetc ($pfichier).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne une chaîne d'un caractère lue dans le fichier pointé par `$pfichier`. La valeur `false` est retournée si le caractère de fin de fichier (`EOF`) est rencontré.

fgetcsv()

- Signature : `fgetcsv ($pfichier, $longueur [, $delimiteur]).`
- Versions : PHP 3 à partir de la 3.0.8, PHP 4, PHP 5.

Cette fonction permet de lire des données ligne par ligne dans un fichier CSV pointé par `$pfichier`. Pour chaque ligne, la fonction renvoie un tableau contenant toutes les données de la ligne. Si le caractère de séparation n'est pas la virgule, il est possible de la spécifier avec le paramètre `$delimiteur`.

Supposez que le fichier *test.csv* contienne les données suivantes :

```
paul,dupont,1982
eric,mullier,1981
marc,rissin,1982
```

Le script suivant permet d'afficher toutes les données ligne par ligne :

```
$ligne = 1;
$pfichier = fopen ("test.csv","r");
while ($tab = fgetcsv ($pfichier, 64))
{
  $num = count ($tab);
```

```
  print "<p> $num champs sur la ligne $row: <br>";
  $ligne++;
  for ($c=0; $c < $num; $c++)
  {
    print $tab[$c] . "<br>";
  }
}
fclose ($pfichier);
```

fgets()

- Signature : `fgets ($pfichier, $ln)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de lire, ligne par ligne, le fichier pointé par `$pfichier`. Le fichier est lu par blocs de `$ln` caractères :

```
$ligne = 1;
$pfichier = fopen ("test.csv","r");
while ($ch = fgets ($pfichier, 64))
{
  print "ligne [$ligne] -> $ch<br>";
  $ligne++;
}
fclose ($pfichier);
```

Avec la fonction `fgetss()`, la ligne reçue est « vidée » de toute balise HTML.

file()

- Signature : `file ($fichier)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction lit le contenu d'un fichier et place chaque ligne dans un tableau qu'elle retourne. Le fichier peut être local ou distant :

```
$tab = file ("http://www.google.fr");
$tab = file ("/sites/monsite/readme.txt");
```

file_exists()

- Signature : `file_exists ($fichier)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne `true` si le fichier `$fichier` existe, sinon `false`. Cette fonction ne peut être utilisée qu'avec les fichiers locaux :

```
if (file_exists($fichier))
{
// actions sur le fichier
}
```

fileatime()

- Signature : `fileatime ($fichier)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la date à laquelle le fichier a été ouvert pour la dernière fois.

Les fonctions `filectime()` et `filemtime()` retournent la date à laquelle le fichier a été modifié pour la dernière fois. Pour `filectime()`, une modification correspond aussi à un changement de permission.

Ces fonctions ne peuvent être utilisées qu'avec des fichiers locaux.

filegroup()

- Signature : `filegroup ($fichier)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne le groupe ID du propriétaire du fichier (ou `false` en cas d'erreur). La valeur retournée est numérique.

Cette fonction n'accepte que des fichiers locaux et ne fonctionne pas sous Windows.

fileinode()

- Signature : `fileinode ($fichier)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction renvoie le numéro inode d'un fichier.

Elle n'accepte que des fichiers locaux et ne fonctionne pas sous Windows.

fileowner()

- Signature : `fileowner ($fichier)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne l'ID du propriétaire du fichier (ou `false` en cas d'erreur). La valeur retournée est numérique.

Cette fonction n'accepte que des fichiers locaux et ne peut être utilisée sous Windows.

fileperms()

- Signature : `fileperms ($fichier)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne les permissions d'un fichier sous forme d'un entier.

Pour voir si un fichier est en lecture publique et pour le groupe, on utilise les opérateurs sur les bits :

```
$permissions = fileperms("fichier.txt");
if (($permissions & 4) && ($permissions & 32)) ...
```

Cette fonction n'accepte que des fichiers locaux et ne peut être utilisée sous Windows.

filesize()

- Signature : `filesize ($fichier)`.
- Versions : PHP 3, PHP 4 à partir de la 4.0.0, PHP 5.

Cette fonction retourne la taille d'un fichier.

Elle n'accepte que des fichiers locaux.

filetype()

- Signature : `filetype ($fichier)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne le type du fichier. Les valeurs possibles sont : `fifo`, `lien`, `répertoire`, `fichier`, etc.

Cette fonction n'accepte que des fichiers locaux.

flock()

- Signature : `flock ($pfichier, $operation).`
- Versions : PHP 3 à partir de la 3.0.7, PHP 4, PHP 5.

Cette fonction permet de bloquer un fichier pointé par `$pfichier`. Les différentes valeurs possibles de la fonction `$operation` sont :

- `LOCK_SH` pour obtenir un blocage partagé (lecture) ;
- `LOCK_EX` pour obtenir un blocage exclusif (écriture) ;
- `LOCK_UN` pour libérer les blocages.

La fonction retourne `true` en cas de succès, sinon `false`.

fopen()

- Signature : `fopen ($fichier, $mode).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet d'ouvrir un fichier et de créer un pointeur sur fichier, qui pourra par la suite être utilisé, par exemple, pour lire ou écrire dans le fichier.

La fonction retourne `false` si l'ouverture a échoué.

Il est possible d'ouvrir un fichier dans différents modes :

- '`r`' ouvre le fichier en lecture et place le pointeur au début du fichier.
- '`r+`' ouvre le fichier en lecture et en écriture et place le pointeur au début du fichier.
- '`w`' ouvre le fichier en écriture et place le pointeur au début du fichier. Si le fichier existe, son contenu est effacé ; s'il n'existe, pas le fichier est créé.

- 'w+' ouvre le fichier en lecture et en écriture et place le pointeur au début du fichier. Si le fichier existe, son contenu est effacé ; s'il n'existe pas, le fichier est créé.
- 'a' ouvre le fichier en écriture et place le pointeur à la fin du fichier. Si le fichier n'existe pas, il est créé.
- 'a+' ouvre le fichier en lecture et en écriture et place le pointeur à la fin du fichier. Si le fichier n'existe pas, il est créé.

La fonction `fopen()` peut ouvrir des fichiers locaux et distants :

```
$pfichier = fopen("http://www.google.fr/index.html","r");
$pfichier = fopen("/sites/monsite/test.txt","w");
// si nous écrivons dans le fichier les données sont
✂ ajoutées au début

$pfichier = fopen("/sites/monsite/test.txt","w");
// si nous écrivons dans le fichier les données sont
✂ ajoutées à la fin
```

fpassthru()

- Signature : `fpassthru ($pfichier).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction lit le fichier pointé par `$pfichier` et affiche le résultat. Seules les données situées en dessous de l'endroit où pointe le fichier sont renvoyées.

La fonction `readfile()` est à préférer si vous souhaitez afficher tout un fichier.

fputs()



Reportez-vous à la fonction `fwrite()`.

fread()

- Signature : `fread ($pfichier, $ln).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de lire dans un fichier pointé par `$pfichier` par blocs de `$ln` octets. La lecture s'arrête quand le caractère de fin de fichier est rencontré (`EOF`).

Certains systèmes comme Windows font une différence entre les fichiers binaires et les fichiers texte. Dans le cas d'un fichier binaire, le mode '`b`' doit être ajouté au mode `$mode` :

```
$fichier = "c:\\tmp\\image.gif";
$pfichier = fopen ($fichier, "rb");
$contenu = fread ($pfichier, filesize ($fichier));
fclose ($pfichier);
```

fscanf()

- Signature : `fscanf ($pfichier, $format).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de lire dans un fichier en utilisant un format, à la manière de `sscanf()`.

Si vous avez un fichier de personnes *.txt* présenté de cette manière :

```
M Paul Dupont [1982]
M Eric Mullier [1981]
M Marc Rissin [1982]
```

Il est possible d'extraire les données ligne par ligne de la manière suivante :

```
$pfichier = fopen ("personnes.txt","r");
while (list ($prenom, $nom, $annee) = fscanf ($pfichier,
✂ "M %s %s [%d]\n"))
{
  print("$nom,$prenom,$annee<br>");
}
fclose($pfichier);
```

fseek()

- Signature : `fseek ($pfichier, $offset [, $origine]).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de déplacer le pointeur sur fichier. La nouvelle position mesurée en octets, par rapport au début du fichier, est obtenue en ajoutant la valeur `$offset` à la position indiquée par `$origine` :

Le paramètre `$origine` peut prendre différentes valeurs :

- SEEK_SET : la nouvelle position vaut $offset octets.
- SEEK_CUR : la nouvelle position vaut la position courante ajoutée à $offset octets.
- SEEK_END : la nouvelle position vaut la position de fin de fichier ajoutée à $offset octets ($offset a donc intérêt à être négatif !).

Si le paramètre $origine n'est pas précisé, la valeur par défaut est SEEK_SET.

La fonction retourne 0 en cas de succès, sinon –1.

fstat()

- Signature : fstat ($pfichier).
- Versions : PHP 4, PHP 5.

Cette fonction permet d'obtenir des informations sur un fichier ouvert. Ces informations sont regroupées dans un tableau. Le tableau contient les valeurs suivantes…

- 0 : volume.
- 1 : inode.
- 2 : nombre de liens.
- 3 : nombre de liens.
- 4 : ID de l'utilisateur propriétaire.
- 5 : ID du groupe propriétaire.
- 6 : type du volume de l'inode.
- 7 : taille en octets.
- 8 : date de dernier accès.
- 9 : date de dernière modification.
- 10 : date du dernier changement.
- 11 : taille de bloc du système pour les entrées et sorties.
- 12 : nombre de blocs alloués.

La fonction stat() permet d'obtenir les mêmes informations en passant un nom de fichier plutôt qu'un identifiant.

ftell()

- Signature : `ftell ($pfichier).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la position du pointeur `$pfichier` et retourne `false` en cas d'erreur.

ftruncate()

- Signature : `ftruncate ($pfichier,$taille).`
- Versions : PHP 4, PHP 5.

Cette fonction prend le pointeur sur fichier `$pfichier` et tronque le fichier à la taille `$taille`.

La fonction retourne `true` en cas de succès, sinon `false`.

fwrite()

- Signature : `fwrite ($pfichier, $ch [, $ln]).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet d'écrire la chaîne de caractères `$ch` dans le fichier pointé par `$pfichier`. Si le paramètre `$ln` est transmis, seuls `$ln` octets seront écrits.

La fonction retourne le nombre d'octets écrits en cas de succès, sinon `–1`.

getcwd()

- Signature : `getcwd ().`
- Versions : PHP 4, PHP 5.

Cette fonction retourne le répertoire courant.

is_dir()

- Signature : `is_dir ($fichier).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne `true` si le fichier `$fichier` est un répertoire.

D'autres fonctions permettent de tester un fichier…

- `is_executable()` : indique qu'il s'agit d'un fichier exécutable.
- `is_file()` : indique qu'il s'agit d'un fichier et non un répertoire.
- `is_link()` : indique un lien symbolique.
- `is_readable()` : le fichier est accessible en lecture.
- `is_writable()`, `is_writeable()` : le fichier est accessible en écriture.
- `is_uploaded_file()` : le fichier est envoyé via la méthode HTTP `POST`.

Ces fonctions n'acceptent que les fichiers locaux.

link()

- Signature : `link ($cible, $lien).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de créer un lien physique.

Elle ne peut pas être utilisée sous Windows.

mkdir()

- Signature : `mkdir ($repertoire, $mode).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de créer un répertoire. Le mode doit être précisé en octal et est modifié par la valeur courante du paramètre `umask` :

```
mkdir ("/sites/monsite/contact", 0700);
// crée le répertoire contact dans le répertoire
✂ /sites/monsite
```

Retourne `true` en cas de succès, sinon `false`.

move_uploaded_file()

- Signature : `move_uploaded_file ($fichier, $destination).`

- Versions : PHP 4 à partir de la 4.0.3, PHP 5.

Cette fonction permet de déplacer un fichier téléchargé en *upload*.

opendir()

- Signature : `opendir ($chemin).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet d'ouvrir un identifiant de répertoire, à l'aide duquel il sera possible de lister les fichiers contenus dans ce répertoire.

Elle retourne `false` en cas d'erreur :

```
// le code ci-dessous permet de lister les fichiers
✂ (répertoires inclus)
// contenus dans le répertoire /tmp
if ($rep = @opendir("/tmp")) {
  while (($nomfichier = readdir($rep)) !== false) {
    echo "$nomfichier\n";
  }
  closedir($rep);
}
```

L'exemple suivant permet de parcourir de manière récursive une arborescence de fichiers :

```
<?php

function parcours($path)
{
  if ($rep = @opendir($path)) {
    while (($fichier = readdir($rep))!== false) {
      if (is_dir($fichier)) {
        if ($fichier=='.' || $fichier=='..') continue;
        parcours("$path/$fichier");
        print("REP : $path/$fichier<br/>\n");
      }
      else print("FICHIER : $path/$fichier<br/>\n");
    }
  }
  closedir($rep);
}

parcours(".");

?>
```

parse_ini_file()

- Signature : `parse_ini_file ($fichier [, $section]).`
- Versions : PHP 4, PHP 5.

Cette fonction retourne un tableau contenant les données stockées dans un fichier *.ini*.

Si `$section` vaut `true`, les sections sont prises en compte.

Le fichier *test.ini* contient :

```
[section1]
val1 = 1
val2 = 2
[section2]
nom = toto
```

```
$tabini = parse_ini_file("test.ini");
print_r($tabini);
/* affiche :
Array
(
    [val1] => 1
    [val2] => 2
    [nom] => toto
)
*/

$tabini = parse_ini_file("test.ini", TRUE);
print_r($tabini);
/* affiche :
Array
(
    [section1] => Array
        (
            [val1] => 1
            [val2] => 2
        )
    [section2] => Array
        (
            [nom] => toto
        )
)
*/
```

pathinfo()

- Signature : `pathinfo ($chemin)`.
- Versions : PHP 4 à partir de la 4.0.3, PHP 5.

Cette fonction retourne un tableau associatif contenant des informations sur le chemin : `dirname`, `basename` et `extension`.

```
$tab = pathinfo("/sites/monsite/index.html");
echo $tab["dirname"] . "\n";  // affiche : /sites/monsite
echo $tab["basename"] . "\n";  // affiche : index.html
echo $tab["extension"] . "\n";  // affiche : html
```

popen()

- Signature : `popen ($commande, $mode)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de créer un processus fils correspondant à l'exécution de la commande `$commande` (les plus érudits peuvent faire l'analogie avec le fork() du langage C).

```
$pfichier = popen ("/bin/ls", "r");
```

La fonction retourne un pointeur sur fichier identique à ceux qui ont été créés avec `fopen()`, à la différence qu'il est unidirectionnel (il est possible soit d'y écrire, soit d'y lire) et qu'il doit être fermé avec la fonction `pclose()`.

Les fonctions `fgets()`, `fwrite()`, etc., peuvent être utilisées avec un tel pointeur.

La fonction retourne `false` en cas d'erreur.

readdir()

- Signature : `readdir ($prep)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne le nom du fichier selon le contenu dans le répertoire identifié par `$prep` :

```
// liste les fichiers contenus dans le répertoire courant
//(les répertoires . et .. sont exclus)
$prep = opendir('.');
while (false !== ($nomfichier = readdir($prep))) {
    if ($nomfichier != "." && $file != "..") {
        echo "$nomfichier\n";
    }
}
closedir($prep);
```

readfile()

- Signature : `readfile ($fichier)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet d'afficher le contenu d'un fichier. Elle peut être utilisée avec des fichiers distants :

```
readfile("ftp://ftp.monserver.com/readme.txt");
// le serveur ftp doit supporter le mode passif !
```

readlink()

- Signature : `readlink ($lien)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la cible d'un lien symbolique.

rename()

- Signature : `rename ($vieuxnom, $nouveaunom)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de renommer un fichier.

```
rename ("fichiers/toto.txt","fichiers/titi.txt");
// renomme toto.txt en titi.txt

rename ("rep1/toto.txt","rep2/toto.txt");
// déplace toto.txt du répertoire rep1 vers celui rep2
```

Retourne `true` en cas de succès, sinon `false`.

rewind()

- Signature : `rewind ($pfichier).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de faire pointer `$pfichier` sur le début du fichier.

rewinddir()

- Signature : `rewinddir ($prep).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de réinitialiser l'identifiant de répertoire et de le faire pointer sur le premier fichier.

rmdir()

- Signature : `rmdir ($repertoire).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet d'effacer un répertoire si celui-ci est vide.

Pour pouvoir effacer un répertoire non vide, il est nécessaire d'écrire une petite fonction, qui se chargera d'aller effacer le contenu du répertoire avant d'appeler la fonction `rmdir()` sur ce même répertoire :

```
function supprim_rep($rep, $flag=0)
{
  $tab_rep = array();
  $tab_fichiers = array();

  $prep = opendir ($rep);
  if (!$prep)
  {
    return null;
  }
  while ($fichier = readdir ($prep))
  {
    if ($fichier == '.' ||  $fichier == '..') continue;
    if (is_dir ("$rep/$fichier"))
   {
      $tab_rep[] = $fichier;
    }
```

```
    else
    {
      $tab_fichiers[] = $fichier;
    }
  }
  $i = 0;
  while ($tab_rep[$i])
  {
    supprim_rep ("$rep/$tab_rep[$i]",1);
    rmdir ("$rep/" . $tab_rep[$i]);
    $i++;
  }
  $i = 0;
  while ($tab_fichiers[$i])
  {
    unlink ("$rep/$tab_fichiers[$i]");
    $i++;
  }
  if ($flag == 0) rmdir ($rep);
  closedir($prep);
  return 1;
}
```

La fonction s'appelle de la manière suivante :

```
supprim_rep($nom_repertoire)
```

Le deuxième paramètre n'est utilisé qu'en interne au sein de la fonction.

set_file_buffer()

- Signature : `set_file_buffer ($pfichier, $buffer)`.
- Versions : PHP 3 à partir de la 3.0.8, PHP 4, PHP 5.

Cette fonction permet de modifier la taille des *buffers* qui sont utilisés pour écrire dans le fichier pointé par `$pfichier`. La taille par défaut est de 8 ko.

En transmettant `0` comme valeur pour `$buffer`, l'écriture n'est plus « bufférisée ». L'avantage est que l'on est sûr que les données sont bien écrites ; l'inconvénient est que l'on réduit considérablement les performances du système.

stat()

- Signature : `stat ($fichier)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction donne des informations sur un fichier, comme `fstat()`.

- `0` : volume.
- `1` : inode.
- `2` : mode de protection de l'inode.
- `3` : nombre de liens.
- `4` : ID de l'utilisateur propriétaire.
- `5` : ID du groupe propriétaire.
- `6` : type du volume de l'inode.
- `7` : taille en octets.
- `8` : date du dernier accès.
- `9` : date de la dernière modification.
- `10` : date du dernier changement.
- `11` : taille de bloc du système pour les entrées et sorties.
- `12` : nombre de blocs alloués.

lstat()

- Signature : `lstat ($fichier)`.
- Versions : PHP 3 à partir de la 3.0.4, PHP 4, PHP 5.

Cette fonction est identique à `stat()`, sauf si le fichier est un lien symbolique. Dans ce cas, c'est le statut de ce lien qui est retourné.

realpath()

- Signature : `realpath ($chemin)`.
- Versions : PHP 4, PHP 5.

Cette fonction retourne le véritable chemin, déterminé à partir du paramètre `$chemin`. Les liens symboliques et les composantes (`.`, `..`) disparaissent.

symlink()

- Signature : `symlink ($cible, $lien)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction crée un lien symbolique.

tempnam()

- Signature : `tempnam ($repertoire, $prefixe)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction crée un fichier avec un nom unique dans le répertoire `$repertoire`. Le nom du fichier est préfixé avec `$prefixe`. Le nom du fichier est retourné.

tmpfile()

- Signature : `tmpfile ()`.
- Versions : PHP 3 à partir de la 3.0.13, PHP 4, PHP 5.

Cette fonction crée un fichier temporaire et retourne un pointeur sur ce fichier :

```
$temp = tmpfile();
fwrite($temp, "ce fichier est temporaire");
fclose($temp);
```

touch()

- Signature : `touch ($fichier [, $date])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de changer la date de modification du fichier `$fichier`. Si la date n'est pas précisée, c'est la date actuelle qui est utilisée.

Si le fichier n'existe pas, il est créé.

La fonction retourne `true` en cas de succès, sinon `false`.

umask()

- Signature : `umask ($masque)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de modifier le paramètre `umask` courant.

unlink()

- Signature : `unlink ($fichier)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de supprimer un fichier.

18.7. L'interface avec MySQL

Dans cette partie, vous utiliserez la norme suivante…

- `$serverbdd` : adresse du serveur de base de données.
- `$utilisateur`, `$motdepasse` : les identifiants permettant de s'identifier auprès du serveur.
- `$bdd` : nom d'une base.
- `$liendb` : identifiant de connexion.
- `$resultat` : identifiant de résultat.

mysql_affected_rows()

- Signature : `mysql_affected_rows ([$liendb])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne le nombre de lignes qui ont été modifiées à la suite de la dernière requête contenant une commande `INSERT`, `UPDATE` ou `DELETE`. Si `$liendb` n'est pas précisé, la dernière connexion ouverte est utilisée :

```
$liendb = mysql_connect('localhost', 'root', '');
mysql_select_db ('test');
$sql = "UPDATE produit SET prix = prix + 1";
mysql_query ($sql);
$n = mysql_affected_rows ($liendb);
```

```
// $n = mysql_affected_rows(); fonctionnerait aussi
echo "$n produits ont été mis à jour";
mysql_close($liendb);
```



Fonction `mysql_affected_rows()` et `UPDATE`

Avec une requête `UPDATE`, MySQL ne met pas à jour les colonnes où l'ancienne valeur est la même que la nouvelle. De ce fait, la fonction `mysql_affected_rows()` ne retourne pas le nombre de lignes qui coïncident avec la requête.

Si la dernière requête a échoué, la fonction retourne `–1`.

mysql_change_user()

- Signature : `mysql_change_user ($utilisateur, $motdepasse[, $bdd , $liendb]])`.
- Versions : PHP 3 à partir de la 3.0.13, PHP 4, PHP 5.

Cette fonction permet de changer l'utilisateur de la connexion courante (ou de la connexion spécifiée par `$liendb`).

Si la base `$bdd` est précisée, elle deviendra la nouvelle base courante de la connexion.

Si l'authentification échoue, la connexion courante reste active.

mysql_close()

- Signature : `mysql_close ([$liendb])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de fermer la connexion courante (ou la connexion identifiée par `$liendb` si le paramètre est précisé).

L'utilisation de `mysql_close()` n'est pas indispensable, dans la mesure où toutes les connexions non persistantes sont automatiquement fermées avec la terminaison du script.

La fonction retourne `true` en cas de succès, sinon `false`.

mysql_connect()

- Signature : mysql_connect ([$serverbdd, $utilisateur, $motdepasse]).
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet d’ouvrir une connexion avec une base de données. Si les paramètres ne sont pas précisés, les valeurs suivantes sont prises par défaut : $serverbdd =’localhost:3306’. $utilisateur est le même utilisateur que le propriétaire du processus et $motdepasse est vide.

```
$liendb = mysql_connect ("localhost", "root", "") or die
✂ ("la connexion à la base a échoué");
print ("connexion réussie");
mysql_close ($liendb);
```

Le paramètre $serverbdd peut contenir un numéro de port (127.0.0.1:3306) ou un chemin vers le socket utilisé pour la connexion (/var/lib/mysql/mysql.sock).

Un nouvel appel à cette fonction, avec les mêmes paramètres, n’ouvre pas de nouvelle connexion, mais retourne le lien de connexion précédemment ouvert.

La fonction mysql_pconnect() ne diffère de mysql_connect() que par le fait que les connexions avec la base sont persistantes. De cette façon, la connexion avec la base n’est pas fermée lorsque le script se termine, et peut ainsi être réutilisée par la suite.

mysql_create_db()

- Signature : mysql_create_db ($bdd [, $liendb]).
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de créer une base de données $bdd :

```
$liendb = mysql_connect(’localhost’, ’root’, ’’);
if (mysql_create_db ("boutique2"))
{
  echo "base créée";
}
else
{
  printf("erreur lors de la création de la base :
  ✂ %s",mysql_error());
}
```

La fonction retourne `true` en cas de succès, sinon `false`.

mysql_data_seek()

- Signature : `mysql_data_seek ($resultat, $ligne)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de déplacer le pointeur contenu dans l'identifiant de résultat vers la ligne `$ligne` :

```
$liendb = mysql_connect('localhost', 'root', '');
mysql_select_db ('test');
$sql = "SELECT nom FROM produit";
$resultat = mysql_query ($sql);
while ($obj = mysql_fetch_object ($resultat)) echo
✂ "$obj->nom - ";
// affiche : tee-shirt - sweat - casquette -
mysql_data_seek ($resultat, 1);  // nous replaçons au
✂ niveau de la ligne 1
print("<br>");
while ($obj = mysql_fetch_object ($resultat)) echo
✂ "$obj->nom - ";
// affiche : sweat - casquette -
mysql_close($liendb);
```

La fonction retourne `true` en cas de succès, sinon `false`.

mysql_db_name()

- Signature : `mysql_db_name ($bdds, $ligne)`.
- Versions : PHP 3 à partir de la 3.0.6, PHP 4, PHP 5.

Cette fonction permet d'obtenir le nom des bases. Le paramètre `$bdds` est le résultat de l'appel à la fonction `mysql_list_dbs()` :

```
mysql_connect('localhost', 'root', '');
$db_list = mysql_list_dbs();
$i = 0;
$nbdd = mysql_num_rows($db_list);
while ($i < $nbdd)
{
    echo mysql_db_name($db_list, $i) . "\n";
    $i++;
}
```

mysql_db_query()

- Signature : `mysql_db_query ($bdd, $requete [, $liendb])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de sélectionner une base de données et d'exécuter, dans la foulée, une requête sur cette base. La connexion courante est utilisée si l'identifiant de connexion `$liendb` n'est pas précisé.

À partir de la version 4.0.6 de PHP, cette fonction ne peut plus être utilisée. Il est nécessaire de passer par `mysql_select_db()`, puis par `mysql_query()`.

La fonction retourne un identifiant de résultat en cas de succès, sinon `false`.

mysql_drop_db()

- Signature : `mysql_drop_db ($bdd [, $liendb])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de supprimer une base.

Elle retourne `true` en cas de succès, sinon `false`.

mysql_errno()

- Signature : `mysql_errno ([$liendb])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne le numéro de l'erreur de la dernière fonction MySQL utilisée. La valeur `0` est utilisée si aucune erreur n'a eu lieu.

mysql_error()

- Signature : `mysql_error ([$leindb])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne le texte d'erreur de la dernière fonction MySQL utilisée. La chaîne vide '' est transmise si aucune erreur n'a eu lieu.

mysql_escape_string()

- Signature : `mysql_escape_string ($str)`.
- Versions : PHP 4 à partir de la 4.0.3.

Cette fonction retourne la chaîne `$str`, dans laquelle les caractères `NUL`, `(\x00)`, `\n`, `\r`, `\`, ', " et `\x1a` sont précédés d'un antislash `\`.

La fonction `addslashes()` se contente de précéder d'un `\` les caractères `NUL`, `'`, " et `\`.

mysql_fetch_array()

- Signature : `mysql_fetch_array ($resultat [, $type])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne un tableau des données lues dans l'identifiant de résultat `$resultat`. En cas d'erreur, la fonction retourne `false`.

Le paramètre optionnel `$type` peut prendre différentes valeurs.

- `MYSQL_ASSOC` : les données sont accessibles par leur nom. Le comportement est alors le même que `mysql_fetch_assoc()`.
- `MYSQL_NUM` : les données sont accessibles par index. Le comportement est alors le même que `mysql_fetch_rox()`.
- `MYSQL_BOTH` : combine les méthodes nom et index. C'est la valeur par défaut.

```
$liendb = mysql_connect('localhost', 'root', '');
mysql_select_db ('test');
$sql = "SELECT reference, nom FROM produit";
$resultat = mysql_query ($sql);
while ($tab = mysql_fetch_array
✂ ($resultat,MYSQL_ASSOC))
{
  echo $tab['nom'];
}
mysql_close($liendb);
```

```
$liendb = mysql_connect('localhost', 'root', '');
mysql_select_db ('test');
$sql = "SELECT reference, nom FROM produit";
$resultat = mysql_query ($sql);
while ($tab = mysql_fetch_array ($resultat,MYSQL_NUM))
{
  echo $tab[1];
}
mysql_close($liendb);
```

mysql_fetch_field()

- Signature : `mysql_fetch_field ($resultat [,$offset])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne un objet contenant des informations sur une colonne.

- `name` : nom de la colonne.
- `table` : nom de la table.
- `max_length` : taille maximale de la colonne.
- `not_null` : `1` si la colonne ne peut pas être `NULL` (attribut `NOT NULL`).
- `primary_key` : `1` si la colonne est une clé primaire (attribut `PRIMARY KEY`).
- `unique_key` : `1` si la colonne est une clé unique (attribut `UNIQUE`).
- `multiple_key` : `1` si la colonne est une clé non unique.
- `numeric` : `1` si la colonne est numérique.
- `blob` : `1` si la colonne est `BLOB`.
- `type` : le type de la colonne.
- `unsigned` : `1` si la colonne est non signée.
- `zerofill` : `1` si la colonne est complétée par des zéros.

```
$liendb = mysql_connect('localhost', 'root', '');
mysql_select_db ('test');
$sql = "SELECT idproduit FROM produit";
$resultat = mysql_query ($sql);
$meta = mysql_fetch_field ($resultat);
echo "<PRE>
blob:          $meta->blob
max_length:    $meta->max_length
```

```
multiple_key: $meta->multiple_key
name:         $meta->name
not_null:     $meta->not_null
numeric:      $meta->numeric
primary_key:  $meta->primary_key
table:        $meta->table
type:         $meta->type
unique_key:   $meta->unique_key
unsigned:     $meta->unsigned
zerofill:     $meta->zerofill
</PRE>";
mysql_close($liendb);
```

mysql_fetch_object()

- Signature : `mysql_fetch_object ($resultat [, $type])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne un objet dont les attributs sont les données lues dans l'identifiant de résultat `$resultat`. La valeur `false` est retournée en cas d'erreur.

```
$liendb = mysql_connect('localhost', 'root', '');
mysql_select_db ('test');
$sql = "SELECT reference, nom FROM produit";
$resultat = mysql_query ($sql);
while ($obj = mysql_fetch_object ($resultat))
{
  echo $obj->nom;
}
mysql_close($liendb);
```

Comme pour `mysql_fetch_array()`, le paramètre `$type` peut être utilisé.

mysql_free_result()

- Signature : `mysql_free_result ($resultat)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction libère toute la mémoire associée à l'identifiant de résultat `$resultat`.

Cette fonction peut être intéressante quand, en plein milieu d'un script, vous procédez à une requête importante sur la base.

La fonction retourne `true` en cas de succès, sinon `false`.

mysql_insert_id()

- Signature : `mysql_insert_id ([$liendb])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne l'`id` généré dans la colonne en `AUTO INCREMENT` par la dernière requête `INSERT`.

```
$liendb = mysql_connect('localhost', 'root', '');
mysql_select_db ('test');
$sql = "INSERT INTO produit (reference) VALUES ('PROD004')";
mysql_query ($sql);
echo "le produit a bien été ajouté à la table, son id est
✂ : " . mysql_insert_id();
mysql_close($liendb);
```

La fonction retourne `0` si la précédente requête n'a pas généré une valeur auto-incrémentée.

mysql_list_dbs()

- Signature : `mysql_list_dbs ([$liendb])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne un identifiant de résultat contenant la liste des bases présentes sur le serveur :

```
$liendb = mysql_connect('localhost', 'root', '');
$list = mysql_list_dbs($liendb);
while ($obj = mysql_fetch_object($list))
{
  echo $obj->Database . "\n";
}
```

mysql_list_fields()

- Signature : `mysql_list_fields ($bdd, $table [, $liendb])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne les colonnes présentes dans une table :

```
$liendb = mysql_connect('localhost', 'root', '');
$colonnes = mysql_list_fields("test", "eleve", $liendb);
$nbcol = mysql_num_fields($colonnes);
for ($i = 0; $i < $nbcol; $i++)
{
  echo mysql_field_name($colonnes, $i) . "\n";
}
```

mysql_list_tables()

- Signature : `mysql_list_tables ($bdd, [$liendb])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet d'obtenir la liste des tables présentes dans la base `$bdd` :

```
$liendb = mysql_connect('localhost', 'root', 'monpasword');
mysql_select_db("test");
$tables = mysql_list_tables("test");
while ( list ($table) = mysql_fetch_array($tables))
{
  echo $table;
}
```

mysql_num_fields()

- Signature : `mysql_num_fields ($resultat)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne le nombre de colonnes contenues dans l'identifiant de résultat.

mysql_num_rows()

- Signature : `mysql_num_rows ($resultat)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne le nombre de lignes contenues dans l'identifiant de résultat :

```
$liendb = mysql_connect('localhost', 'root', '');
mysql_select_db ('test');
$sql = "SELECT idproduit FROM produit WHERE prix ≥ 20 ";
```

```
$resultat = mysql_query ($sql);
echo "cette table contient " . mysql_num_rows($resultat) .
✂ " produits ayant un prix ≥ 20";
mysql_close($liendb);
```

mysql_query()

- Signature : `mysql_query ($requete [, $liendb])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de transmettre une requête à la base. Seule la commande `SELECT` permet d'obtenir un identifiant de résultat.

La fonction retourne `false` en cas d'échec.

Il est désormais impossible d'envoyer deux requêtes en même temps à la base. Cela évite de nombreux problèmes de sécurité :

```
mysql_query ("SELECT * FROM produit; DROP test");
// impossible
```

mysql_result()

- Signature : `mysql_result ($resultat, $ligne [, $colonne])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de récupérer le contenu d'une donnée se trouvant dans l'identifiant de résultat. Le paramètre `$colonne` peut, à la fois, être un index ou un nom de colonne.

mysql_select_db()

- Signature : `mysql_select_db ($bdd [, $liendb])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet d'initialiser la base courante.

Elle retourne `true` en cas de succès, sinon `false`. Toutes les requêtes effectuées sur le serveur avec la fonction `mysql_query()` sont faites sur la base courante.

mysql_get_client_info()

- Signature : `mysql_get_client_info().`
- Versions : PHP 4 à partir de la 4.0.5, PHP 5.

Cette fonction retourne la version de la librairie MySQL cliente :

```
echo mysql_get_client_info();
// affiche par exemple : 3.23.39
```

mysql_get_host_info()

- Signature : `mysql_get_host_info ([$liendb]).`
- Versions : PHP 4 à partir de la 4.0.5, PHP 5.

Cette fonction retourne le type de connexion :

```
echo mysql_get_host_info();
//affiche par exemple : Localhost via UNIX socket
```

mysql_get_proto_info()

- Signature : `mysql_get_proto_info ([$liendb]).`
- Versions : PHP 4 à partir de la 4.0.5, PHP 5.

Cette fonction retourne la version du protocole :

```
echo mysql_get_proto_info();  // affiche par exemple : 10
```

mysql_get_server_info()

- Signature : `mysql_get_server_info().`
- Versions : PHP 4 à partir de la 4.0.5, PHP 5.

Cette fonction retourne la version du serveur :

```
echo mysql_get_server_info();
// affiche par exemple : 3.23.41
```

18.8. Les images

Pour toutes les fonctions qui suivent, l'origine de l'image (0, 0) est située dans l'angle supérieur gauche de l'image.

`$img` correspond à un identifiant d'image, créé par exemple avec `ImageCreate()`.

getimagesize()

- Signature : `getimagesize ($fichier)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne un tableau contenant les dimensions d'une image. Le fichier peut être local ou distant :

```
$tab = GetImageSize ("http://www.php.net/gifs/logo.gif");
echo "largeur : " . $tab[0];  // affiche 130
echo "longueur : " . $tab[1];  // affiche 67
```

Deux autres éléments sont présents dans le tableau :

- le type de l'image (`1` représente le format GIF, `2` JPEG, `3` PNG, `4` SWF, `5` PSD, `6` BMP) ;
- une chaîne du type `height=xxx width=xxx`, qui peut être utilisée directement dans une balise `<IMG>`.

Image2WBMP()

- Signature : `Image2WBMP ($img [, $fichier])`.
- Versions : PHP 4 à partir de la 4.0.5, PHP 5.

Cette fonction permet de convertir une image au format WBMP (format d'image pour le Wap, par exemple).

Si le paramètre `$fichier` n'est pas transmis, le contenu est affiché directement à l'écran.

ImageAlphaBlending()

- Signature : `ImageAlphaBlending($img, $mode)`.
- Versions : PHP 4 à partir de la 4.0.6, PHP 5.

Cette fonction permet de passer en mode alpha pour les fonctions d'affichage. Cela signifie que les couleurs disposeront d'une dimension en

plus : la transparence. Le paramètre `$mode` doit prendre la valeur `true` pour passer en mode de transparence. Cette fonction doit être utilisée avec des images créées en mode True Color. La fonction `ImageCreateTruecolor($dimX,$dimY)` permet de créer une telle image.

Listing 18-1 : Transparence dans les images

```
<?
$imOrigine = @imagecreatefrompng ("logo.png");
$im = ImageCreateTrueColor(276,110);
// copie de l'image dans une image en True Color
ImageCopy($im,$imOrigine,0,0,0,0,276,110);
ImageFilledRectangle($im,0,0,92,110,0x00ff1111);
// passage en mode transparence
ImageAlphaBlending($im,true);
ImageFilledRectangle($im,92,0,184,110,0x20ff1111);
ImageFilledRectangle($im,184,0,276,110,0x60ff1111);
Header('Content-Type: image/png');
ImagePNG($im);
?>
```

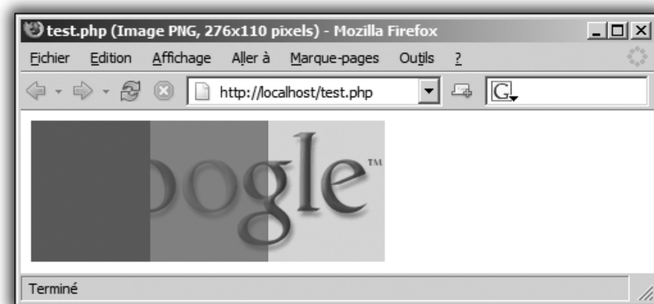


Figure 18.2 : *Exemple de l'utilisation des transparences*

ImageArc()

- Signature : `ImageArc ($img, $cx, $cy, $larg, $haut, $deb, $fin, $coul).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de dessiner une ellipse partielle, centrée sur le point de coordonnées (`$cx, $cy`) dans l'image identifiée par `$img`. L'ellipse a pour largeur `$larg` et pour hauteur `$haut`. Les points de départ et d'arrivée sont spécifiés par `$deb` et `$fin`, et sont indiqués en degrés (dans le sens des aiguilles d'une montre).

$coul est un identifiant de couleur.

```
<?php
Header ("Content-type: image/png");
$img = ImageCreate (250, 250);

$blanc = ImageColorAllocate ($img, 255, 255, 255);
$noir = ImageColorAllocate ($img, 0, 0, 0);
ImageArc ($img, 125, 125, 120, 90, 0, 230, $noir);
// l'ellipse partielle

ImageArc ($img, 125, 125, 50, 50, 0, 360, $noir);
// le cercle central

ImagePng ($img);
ImageDestroy ($img);
?>
```

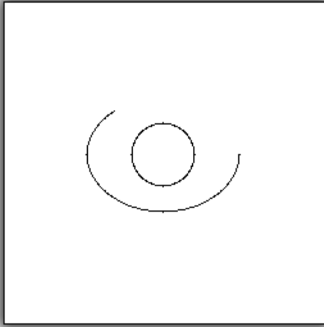


Figure 18.3 :
Une ellipse incomplète et un cercle

ImageChar()

- Signature : `ImageChar ($img, $font, $x, $y, $c, $coul)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction dessine dans l'image identifiée par `$img` le premier caractère de la chaîne `$c`. Le point (`$x`,`$y`) correspond à l'angle supérieur gauche du caractère. Si la valeur de `$font` est `1`, `2`, `3`, `4` ou `5`, une des polices par défaut sera utilisée.

La fonction `ImageCharUp()` permet d'afficher le caractère verticalement.

ImageColorAllocate()

- Signature : `ImageColorAllocate ($img, $rouge, $ver, $bleu).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de créer un nouvel identifiant de couleur, qui pourra être utilisé dans l'image `$img`. Les paramètres `$rouge`, `$vert`, `$bleu` sont compris entre `0` et `255` :

```
$blanc = ImageColorAllocate ($img, 255, 255, 255);
$noir = ImageColorAllocate ($img, 0, 0, 0);
```

En multipliant le nombre de couleurs allouées, vous obtiendrez facilement un dégradé :

Listing 18-2 : Dégradé

```
<?
$im = ImageCreate(300,256);
for($r=0; $r<256; $r++) {
$col = ImageColorAllocate($im,$r,0,0);
ImageLine($im, 0,$r, 100, $r, $col);
}
for($g=0; $g<256; $g++) {
$col = ImageColorAllocate($im,0,$g,0);
ImageLine($im, 100,255-$g, 200, 255-$g, $col);
}
for($b=0; $b<256; $b++) {
$col = ImageColorAllocate($im,0,0,$b);
ImageLine($im, 200,$b, 300, $b, $col);
}
Header('Content-Type: image/png');
ImagePNG($im);
?>
```

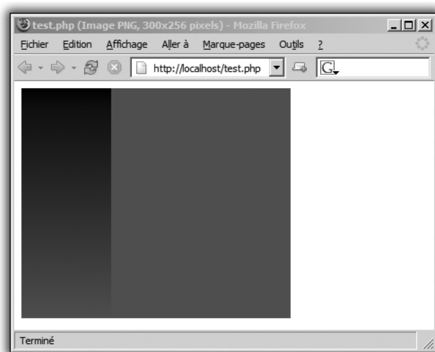


Figure 18.4 : *Affichage du dégradé*

ImageColorDeAllocate()

- Signature : `ImageColorDeAllocate ($img, $coul).`
- Versions : PHP 3 à partir de la 3.0.6, PHP 4, PHP 5.

Cette fonction permet de désattribuer une couleur d'une image.

ImageColorAt()

- Signature : `ImageColorAt ($img, $x, $y).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de récupérer l'identifiant de couleur du pixel de coordonnées (`$x`, `$y`).

ImageColorClosest()

- Signature : `ImageColorClosest ($img, $rouge, $vert, $bleu).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne l'identifiant de couleur de la palette de l'image `$img`, qui est la plus proche de la couleur définie par les composantes (`$rouge`, `$vert`, `$bleu`).

ImageColorExact()

- Signature : `ImageColorExact ($img, $rouge, $vert, $bleu).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne l'identifiant de couleur spécifié par les composantes (`$rouge`, `$vert`, `$bleu`).

La valeur `–1` est retournée si la couleur n'est pas présente dans la palette.

ImageGammaCorrect()

- Signature : `ImageGammaCorrect ($img, $inputgamma, $outgamma).`
- Versions : PHP 3 à partir de la 3.0.13, PHP 4, PHP 5.

Cette fonction applique une correction gamma à l'image identifiée par `$img`.

ImageColorsTotal()

- Signature : `ImageColorsTotal ($img).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne le nombre de couleurs présentes dans la palette de l'image.

ImageColorTransparent()

- Signature : `ImageColorTransparent ($img, $coul).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de définir quel est l'identifiant de couleur qui correspond au transparent dans l'image.

ImageCopy()

- Signature : `ImageCopy ($dest_img, $orig_img, $dest_x, $dest_y, $orig_x, $orig_y, $orig_larg, $orig_haut).`
- Versions : PHP 3 à partir de la 3.0.6, PHP 4, PHP 5.

Cette fonction copie une partie d'une image d'origine (`$orig_img`) vers une image de destination (`$dest_img`). Pour l'image de départ, les coordonnées de l'angle supérieur gauche de la partie à extraire sont précisées, ainsi que la largeur et la hauteur de cette même partie. Pour l'image de destination, seules les coordonnées de l'endroit où va être copiée la zone sont précisées.

ImageCreate()

- Signature : `ImageCreate ($larg, $haut).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction crée une image de largeur `$larg` et de longueur `$long`. Un identifiant d'image est retourné.

ImageCreateFromGIF()

- Signature : `ImageCreateFromGIF ($fichier).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de créer un identifiant d'image à partir d'une image GIF (présente localement ou à distance).

D'autres fonctions permettent de faire exactement la même chose pour d'autres formats : `ImageCreateFromJPEG()`, `ImageCreateFromPNG()`, `ImageCreateFromWBMP()`, `ImageCreateFromString()` (l'image est créée à partir d'une chaîne de caractères), `ImageCreateFromXBM()`, `ImageCreateFromXPM()`.

ImageDestroy()

- Signature : `ImageDestroy ($img).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de détruire toute la mémoire associée à l'identifiant d'image `$img`.

ImageFill()

- Signature : `ImageFill ($img, $x, $y, $coul).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de remplir une région de l'image avec la couleur reconnue par l'identifiant de couleur `$coul`. L'angle supérieur gauche de la région a pour coordonnées (`$x`, `$y`).

ImageFilledPolygon()

- Signature : `ImageFilledPolygon ($img, $tabpoints, $nbpoints, $coul).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de dessiner dans `$img` un polygone plein, de couleur `$coul`. Le tableau doit être conçu de la manière suivante : `$tabpoints[0]` correspond à `x0`, `$tabpoints[1]` à `y0`,

`$tabpoints[2]` à `x1`, `$tabpoints[3]` à `y1`, etc. Le paramètre `$nbpoints` contient le nombre de sommets du polygone.

ImageFilledRectangle()

- Signature : `ImageFilledRectangle ($img, $x1, $y1, $x2, $y2, $coul)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de dessiner un rectangle plein.

ImageFillToBorder()

- Signature : `ImageFillToBorder ($img, $x, $y, $coulbord, $coul)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction remplit avec la couleur `$coul` toute la région limitée par la couleur `$coulbord`. Le point de départ est (`$x`,`$y`) :

```
<?php
Header ("Content-type: image/png");
$img = ImageCreate (250, 250);

$gris = ImageColorAllocate ($img, 204, 204, 204);
$bleu = ImageColorAllocate ($img, 0, 0, 255);
$bleugris = ImageColorAllocate ($img, 173, 173, 205);
ImageLine ($img, 0, 150, 150, 0, $bleu);
imagefilltoborder ($img, 10, 10, $bleu, $bleugris);
ImagePng ($img);
ImageDestroy ($img);
?>
```

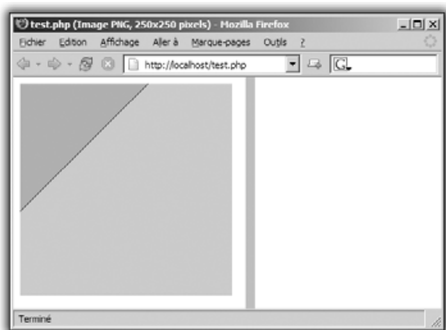


Figure 18.5 : *Remplissage depuis le point (10, 10)*

Si vous choisissez, maintenant, le point de coordonnées (200, 200), c'est l'autre zone qui est remplie avec la couleur gris-bleu.

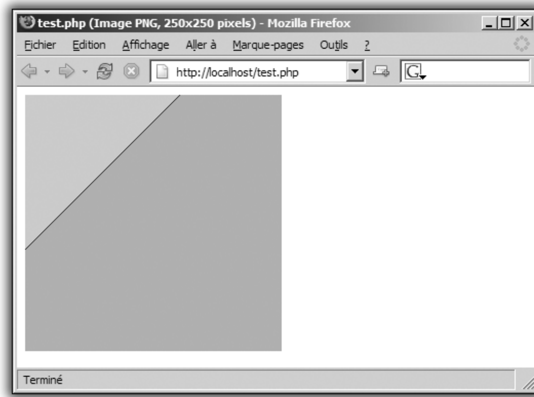


Figure 18.6 : *Remplissage depuis le point (200, 200)*

ImageFontHeight()

- Signature : `ImageFontHeight ($font).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la hauteur (en pixels) d'un caractère de la fonte `$font`.

ImageFontWidth()

- Signature : `ImageFontWidth ($font).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la largeur (en pixels) d'un caractère de la fonte `$font`.

ImageGIF()

- Signature : `ImageGIF ($img [, $fichierimage]).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de générer le contenu de l'image identifiée par `$img` au format GIF. Si le paramètre `$fichierimage` n'est pas précisé, le contenu est affiché directement à l'écran.

Cette fonction n'est plus disponible avec les versions de la libGD supérieures ou égales à 1.6.

D'autres formats peuvent être générés avec les fonctions suivantes…

- `ImagePNG()` : génération au format PNG.
- `ImageJPEG()` : génération au format JPEG.
- `ImageWBMP()` : génération au format WBM.

```
Header ("Content-type: image/png");
$img = ImageCreate (180, 125);
$bleu = ImageColorAllocate ($img, 0, 0, 255);
$blanc = ImageColorAllocate ($img, 255, 255, 255);
// génération d'un carré blanc dans un rectangle blanc
ImageFilledRectangle ($img, 50, 50, 105, 105, $blanc);
ImagePng ($img);
```

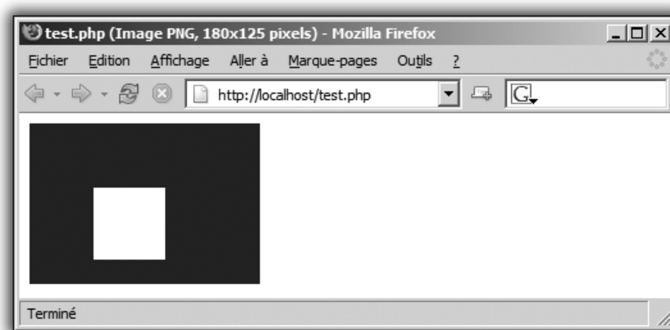


Figure 18.7 : *Exemple d'une image PNG générée à la volée*

ImageInterlace()

- Signature : `ImageInterlace ($img [, $interlace])`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet d'activer ou de désactiver le bit d'entrelacement. Si le paramètre `$interlace` vaut `1`, l'image sera interlacée ; s'il vaut `0`, elle ne le sera pas.

Si le bit d'entrelacement est activé et que le format de l'image soit JPEG, l'image sera créée en tant que JPEG progressif.

ImageLine()

- Signature : `ImageLine ($img, $x1, $y1, $x2, $y2, $coul)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de tracer une ligne de couleur `$coul` entre un point d'origine (`$x1`,`$y1`) et un point final (`$x2`,`$y2`).

ImageLoadFont()

- Signature : `ImageLoadFont ($fichierfont)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de charger un fichier de police de caractères. La fonction retourne un identifiant de fonte.

ImagePaletteCopy()

- Signature : `ImagePaletteCopy ($imgsrc, $imgdest)`.
- Versions : PHP 4, PHP 5.

Cette fonction permet de copier la palette d'une image `$imgsrc` vers une image `$imgdest`.

ImagePolygon()

- Signature : `ImagePolygon ($img, $tabpoints, $nbpoints, $coul)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de dessiner dans `$img` un polygone de couleur `$coul`. Le tableau doit être conçu de la manière suivante : `$tabpoints[0]` correspond à `x0`, `$tabpoints[1]` à `y0`, `$tabpoints[2]` à `x1`, `$tabpoints[3]` à `y1`, etc. Le paramètre `$nbpoints` contient le nombre de sommets du polygone :

```
<?php
Header ("Content-type: image/png");
$img = ImageCreate (250, 250);
// couleur du fond de l'image
$gris = ImageColorAllocate ($img, 204, 204, 204);
$bleu = ImageColorAllocate ($img, 0, 0, 255);
// définition des sommets
$tab_sommets = array (
"20",    // abscisse point 1
"100",  // ordonnée point 1
"80",    // abscisse point 2
"20",    // ordonnée point 2
"180",  // abscisse point 3
"20",    // ordonnée point 3
"230",  // abscisse point 4
"100",  // ordonnée point 4
"150",  // abscisse point 5
"220",  // ordonnée point 5
);
ImagePolygon ($img, $tab_sommets, 5, $bleu);
ImagePng ($img);
ImageDestroy ($img);
?>
```

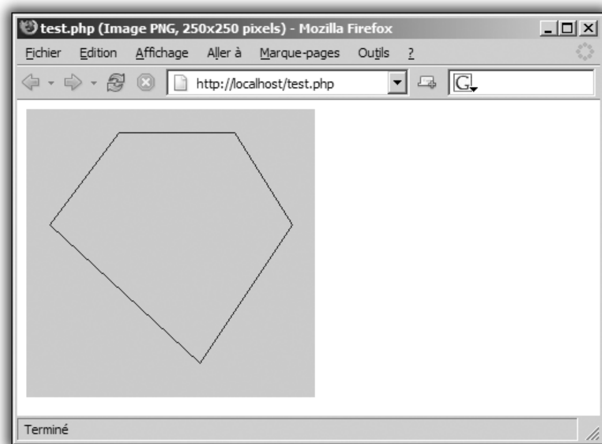


Figure 18.8 : *Fonction ImagePolygon()*

ImageRectangle()

- Signature : `ImageRectangle ($img, $x1, $y1, $x2, $y2, $coul)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de dessiner un rectangle.

ImageSetPixel()

- Signature : `ImageSetPixel ($img, $x, $y, $coul).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de changer la couleur d'un pixel.

ImageString()

- Signature : `ImageString ($img, $font, $x, $y, $str, $coul).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de dessiner la chaîne de caractères `$str` dans l'image `$img`. Si `$font` vaut `1`, `2`, `3`, `4` ou `5`, une police par défaut est utilisée.

`ImageStringUp()` permet de dessiner la chaîne en vertical.

ImageSX()

- Signature : `ImageSX ($img).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la largeur d'une image représentée par l'identifiant d'image `$img`.

`ImageSY()` retourne la hauteur.

FonctionImageTTFBBox()

- Signature : `ImageTTFBBox ($taille, $angle, $fichierfont, $str).`
- Versions : PHP 3 à partir de la 3.0.1, PHP 4, PHP 5.

Cette fonction retourne un tableau contenant les coordonnées de la région qui entoure la zone de texte dessinée avec les paramètres suivants…

- `$taille` : taille de la fonte.
- `$angle` : angle, en degrés, fait par le texte par rapport à l'horizontal.
- `$fichierfont` : chemin d'accès à la fonte.
- `$str` : texte à dessiner.

Les éléments du tableau sont les suivants…

- `0` : abscisse de l'angle inférieur gauche.
- `1` : ordonnée de l'angle inférieur gauche.
- `2` : abscisse de l'angle inférieur droit.
- `3` : ordonnée de l'angle inférieur droit.
- `4` : abscisse de l'angle supérieur droit.
- `5` : ordonnée de l'angle supérieur droit.
- `6` : abscisse de l'angle supérieur gauche.
- `7` : ordonnée de l'angle supérieur gauche.

ImageTTFText()

- Signature : `ImageTTFText ($img, $taille, $angle, $x, $y, $coul, $fichierfont, $str).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de dessiner la chaîne de caractères `$str` dans l'image `$img`. Le point de coordonnées (`$x`, `$y`) correspond à l'angle inférieur gauche. Le nom, la couleur et la taille de la fonte peuvent être précisés avec les paramètres `$taille`, `$coul`, et `$fichierfont`. L'angle (en degrés) que fait le texte avec l'horizontal est précisé par `$angle` (la valeur `90` permet d'afficher le texte verticalement) :

```
Header ("Content-type: image/png");
$img = ImageCreate (180, 125);
$blue = ImageColorAllocate ($img, 0, 0, 255);
$white = ImageColorAllocate ($img, 255, 255, 255);
ImageTTFText ($img, 8, 0, 10, 20, $white, "./verdana.ttf",
              "test");
ImagePng ($img);
```

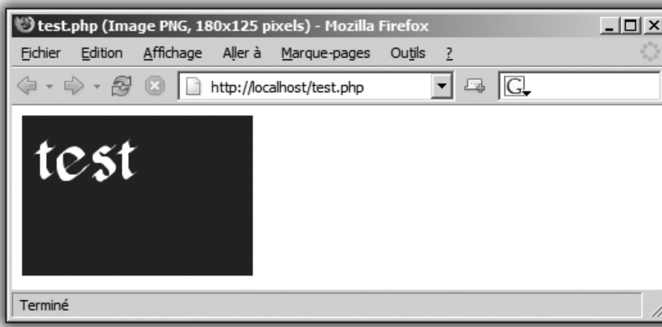


Figure 18.9 : *Image contenant du texte générée à la volée*

La fonction retourne les coordonnées de la région entourant le texte.



Reportez-vous à la fonction `ImageTTFBBox()`.

ImageTypes()

- Signature : `ImageTypes()`.
- Versions : PHP 3 CVS seulement, PHP 4 à partir de la 4.0.2, PHP 5.

Cette fonction permet de savoir quels formats d'image sont pris en charge. Cette fonction renvoie un champ de bits.

Les différents paramètres de format possibles sont : `IMG_GIF`, `IMG_JPG`, `IMG_PNG`, `IMG_WBMP`.

```
if (ImageTypes() & IMG_PNG)  echo "le format PNG est supporté";
```

JPEG2WBMP()

- Signature : `JPEG2WBMP ($fichierjpeg, $fichierwbm, $haut, $larg)`.
- Versions : PHP 4 à partir de la 4.0.5, PHP 5.

Cette fonction permet de convertir une image JPEG en une image WBM de largeur `$larg` et de hauteur `$haut`.

`PNG2WBMP()` fonctionne sur le même modèle.

18.9. Les variables

empty ()

- Signature : `empty ($var)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne `FALSE` si la variable `$var` est définie ou bien si sa valeur est différente de `0`.

gettype ()

- Signature : `gettype ($var)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne le type de la variable `$var`.

Les différents types sont : `boolean`, `integer`, `double`, `string`, `array`, `object`, `resource`, `NULL`.

Les fonctions de test du type sont les suivantes…

- `is_array ($var)` : permet de tester si la variable `$var` est un tableau.
- `is_bool ($var)` : teste si `$var` est un booléen.
- `is_double()` : alias de la fonction `is_float()`.
- `is_float ($var)` : teste si `$var` est un nombre à virgule (flottant).
- `is_int ($var)` : teste si `$var` est un entier.
- `is_integer()` : alias de la fonction `is_int()`.
- `is_long()` : alias de la fonction `is_int()`.
- `is_null ($var)` : teste si `$var` a une valeur égale à `NULL`.
- `is_numeric ($var)` : teste si `$var` est un nombre ou une chaîne de caractères contenant un nombre.
- `is_object ($var)` : teste si `$var` est un objet.
- `is_real()` : alias de la fonction `is_float()`.
- `is_resource ($var)` : teste si `$var` est de type ressource (pointeur sur fichier, identifiant de connexion, etc.).

- is_scalar ($var) : teste si $var est un scalaire (c'est-à-dire un entier, un nombre flottant, un booléen ou une chaîne de caractères).
- is_string ($var) : teste si $var est une chaîne de caractères.

```
<?php

$x = '1';

if (is_string($x)) echo "string"; // affiche string
if (is_numeric($x)) echo "numeric"; // affiche
✂ numeric
if (is_bool($x)) echo "booléen";
if (is_double($x)) echo "double";
if (is_float($x)) echo "float";
if (is_int($x)) echo "int";


$x = '1' + 0;

if (is_string($x)) echo "string";
if (is_numeric($x)) echo "numeric"; // affiche
✂ numeric
if (is_bool($x)) echo "booléen";
if (is_double($x)) echo "double";
if (is_float($x)) echo "float";
if (is_int($x)) echo "int"; // affiche int

?>
```

isset()

- Signature : isset ($var).
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction renvoie true si la variable $var est définie.

print_r()

- Signature : print_r($tab).
- Versions : PHP 4, PHP 5.

Cette fonction permet d'afficher le contenu d'un tableau.

Il est intéressant d'entourer la fonction des balises <pre> et </pre> pour rendre l'affichage du tableau plus lisible :

```
$tab = array ("a" => array ("b","c"), "d" => 1, "e" =>
✂ array (2,3));
echo "<pre>";
print_r($tab);
echo "</pre>";
/* affiche :
Array
(
    [a] => Array
        (
            [0] => b
            [1] => c
        )
    [d] => 1
    [e] => Array
        (
            [0] => 2
            [1] => 3
        )
)
*/
```

settype()

- Signature : `settype ($var, $type).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de forcer le type d'une variable :

```
$var1 = "5toto";  // string
$var2 = true;   // boolean

settype($var1, "integer");
// $var1 vaut maintenant 5   (integer)

settype($var2, "string");
// $var2 vaut maintenant "1" (string)
```

unset()

- Signature : `unset ($var).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de supprimer une variable.

Elle peut être utilisée pour effacer un élément d'un tableau :

```
$tab = array ("a" => array ("b","c"), "d" => 1, "e" => 2);
unset($tab['d']);
echo "<pre>";
print_r($tab);
echo "</pre>";
/* affiche :
Array
(
    [a] => Array
        (
            [0] => b
            [1] => c
        )
    [e] => 2
)
*/
```

Si le tableau est de type scalaire, prenez garde à ne pas utiliser une boucle pour parcourir tout le tableau :

```
$tab = array ('a','b','c','d');
unset($tab[2]);
$i = 0;
while ($tab[$i])
{
  echo "$tab[$i] - ";
  $i++;
}
// affiche a - b -
// en effet l'élément 2 n'existe plus, $tab[2] est
// équivalent à false
echo "<pre>";
print_r($tab);
echo "</pre>";
/* affiche
Array
(
    [0] => a
    [1] => b
    [3] => d
)
*/
```

var_dump()

- Signature : `var_dump ($var1 [,$var2…])`.
- Versions : PHP 3 à partir de la 3.0.5, PHP 4, PHP 5.

Cette fonction permet d'afficher des informations sur une ou plusieurs variables :

```
$var1 = 10.5;
$var2 = true;
var_dump($var1,$var2);
/* affiche :
float(10.5)
bool(true)
*/
```

18.10. La configuration PHP

dl()

- Signature : `dl ($extension)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de charger une extension :

```
dl("xml.so");  // pour charger l'extension XML sous UNIX
dl("xml.dll");  // pour charger l'extension XML sous
✂ windows
```

getenv()

- Signature : `getenv ($var)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de récupérer le contenu d'une variable d'environnement :

```
echo getenv ("HOSTTYPE");  // affiche par exemple : i386
```

get_cfg_var()

- Signature : `get_cfg_var ($var)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de récupérer le contenu d'une variable de configuration PHP :

```
echo get_cfg_var ("file_uploads");
// affiche par exemple : 1 , il est donc possible
// d'uploader des fichiers
```

get_current_user()

- Signature : `get_current_user()`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne le nom du propriétaire du script en cours d'exécution :

```
echo get_current_user();  // affiche par exemple : nobody
```

Avec la fonction `getmyuid()`, c'est le *USER ID* du propriétaire qui est retourné :

```
echo getmyuid();  // affiche par exemple : 99
```

get_defined_constants()

- Signature : `get_defined_constants()`.
- Versions : PHP 4 à partir de la 4.0.7RC1, PHP 5.

Cette fonction retourne un tableau contenant toutes les constantes définies (y compris celles qui sont définies par les extensions).

get_extension_funcs()

- Signature : `get_extension_funcs ($module)`.
- Versions : PHP 4, PHP 5.

Cette fonction retourne la liste des fonctions présentes dans le module `$module` :

```
print_r (get_extension_funcs ("gd"));
```

getmygid()

- Signature : `getmygid()`.
- Versions : PHP 4 à partir de la 4.0.7RC1, PHP 5.

Cette fonction retourne le *GROUP ID* du propriétaire du script courant.

get_loaded_extensions()

- Signature : `get_loaded_extensions()`.
- Versions : PHP 4, PHP 5.

Cette fonction retourne un tableau contenant le nom de toutes les extensions chargées.

get_magic_quotes_gpc()

- Signature : `get_magic_quotes_gpc()`.
- Versions : PHP 3 à partir de la 3.0.6, PHP 4, PHP 5.

Cette fonction permet de savoir si les *magic quotes* fonctionnent.

L'abréviation *gpc* (get/post/cookie) signifie que les variables provenant d'un formulaire ou d'un cookie sont échappées.

```
if (get_magic_quotes_gpc())
  echo "le système utilise les 'magic quotes'";
else
  echo "le système n'utilise pas le système des
  'magic quotes'";
```

ini_alter()

- Signature : `ini_alter ($var, $val)`.
- Versions : PHP 4, PHP 5.

Cette fonction permet de modifier le contenu d'une variable de configuration.

ini_get()

- Signature : `ini_get ($var)`.
- Versions : PHP 4, PHP 5.

Cette fonction retourne le contenu d'une variable de configuration :

```
echo ini_get ("file_uploads")
```

ini_restore()

- Signature : `ini_restore ($var)`.
- Versions : PHP 4, PHP 5.

Cette fonction permet de remettre une variable de configuration à sa valeur initiale.

ini_set()



Reportez-vous à la fonction `init_alter()`.

phpversion()

- Signature : `phpversion()`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne la version de PHP.

putenv()

- Signature : `putenv ($str)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet d'initialiser une variable d'environnement :

```
putenv ("UNIQID=$uniqid");
```

18.11. Fonctions diverses

constant()

- Signature : `constant($cst)`.
- Versions : PHP 4 à partir de la 4.0.4, PHP 5.

Cette fonction retourne la valeur de la constante `$cst` :

```
define("MAXINFO",200);
echo MAXINFO;
echo constant("MAXINFO");
// même résultat que précédemment
```

define()

- Signature : `define ($cst, $val)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de définir une constante.

defined()

- Signature : `defined ($cst)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction retourne `true` si la constante `$cst` existe.

die()



Reportez-vous à la fonction `exit()`.

eval()

- Signature : `eval ($str)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet d'évaluer une expression :

```
eval ("\$n = 1;");
eval ("\$ch = \"hello\";");
echo $n;  // affiche : 1
echo $ch;  // affiche : hello
```

exit()

- Signature : `exit ($status)`.
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction affiche la variable `$status` et arrête l'exécution du script :

```
if ($n != 2)
   exit("valeur non conforme");
$pfichier = fopen ($fichier, 'r')
            or exit("le fichier ne peut être ouvert");
```

highlight_file()

- Signature : `highlight_file ($fichier)`.
- Versions : PHP 4, PHP 5.

Cette fonction permet d'afficher le contenu d'un script en mettant le code en couleur :

```
echo "voici le contenu du script : graph.php<hr/>";
highlight_file("graph.php");
```

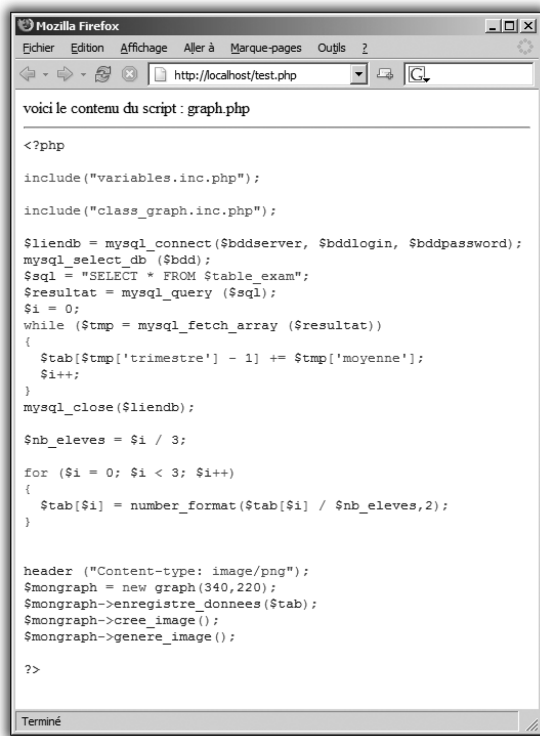


Figure 18.10 : *Affichage colorisé du contenu du script graph.php*

La fonction `highlight_string()` ne colorise que la chaîne transmise en paramètre :

```
highlight_string ("<?php \$str = \"toto\"; ?>");
```

sleep()

- Signature : `sleep ($n).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet d'arrêter l'exécution d'un script pendant `$n` secondes.

La fonction `usleep()` prend en argument un nombre de millisecondes.

uniqid()

- Signature : `uniqid ($refixe).`
- Versions : PHP 3, PHP 4, PHP 5.

Cette fonction permet de générer une valeur unique. Si le paramètre `$prefixe` est transmis la valeur est précédée de `$prefixe` :

```
echo uniqid("");
// affiche par exemple : 3c05f028546e6

echo uniqid("CODE");
// affiche par exemple : CODE3c05f028594cb
```

Chapitre 19

Annexes

Webographie 650
PHP 654
MySQL 673
Les caractères HTML spéciaux 679
Les feuilles de styles : CSS 683

19.1. Webographie

L'arrivée du Web a radicalement modifié la vie du développeur. Rarissimes sont désormais les situations où vous ne pouvez trouver une réponse à un problème sur le Web. Bien souvent, le code répondant à votre question pourra même être téléchargé gratuitement.

Ce chapitre répertorie quelques-uns des sites phares autour de PHP et, plus généralement, des applications en ligne.

PHP

Les sites officiels

- **www.php.net** : site officiel du langage PHP. L'espace de téléchargement de PHP fournit des sources ou des binaires pour différentes plateformes. Un espace est consacré à la documentation la plus à jour. Possibilité de la télécharger dans différents formats : HTML, PDF, HLP (fichiers d'aide Windows).
- **http://snaps.php.net** : permet de récupérer les versions les plus récentes de PHP (généralement compilées pendant la nuit). Ce sont des versions de développement par définition instables.
- **www.zend.com** : site de la société Zend. Zone de téléchargement : la librairie Zend Optimizer et une version de démonstration de l'outil de développement Zend IDE. Toutes les semaines, deux articles présentant les dernières avancées au sein de PHP et du Zend Engine sont mis en ligne. Beaucoup d'articles, de codes et de documentation, toujours particulièrement bien mis en pages.
- **oss.backendmedia.com/PhP60** : un état d'avancement du futur PHP6.

Les sites français

- **www.nexen.net** : la documentation de PHP en français.
- **www.phpindex.com**, **www.phpinfo.net**, **www.phpfrance.com** : documentation, FAQ, exemples, liens.
- **www.commentcamarche.net** : site de vulgarisation informatique d'une richesse rare.
- **www.ofphp.com** : l'observatoire français du PHP.

Les documentations et articles

- **www.phpbuilder.com** : beaucoup d'articles de qualité. Forums très actifs.
- **www.devshed.com/c/b/PHP** : articles et exemples.
- **www.alt-php-faq.org**, **www.faqts.com/knowledge_base/index.phtml/fid/51** : des sites qui permettent de trouver des réponses à des problèmes plus ou moins courants (ce sont, en fait, des FAQ).
- **www.linuxdoc.org** : site central pour toute la documentation de l'open source. On y trouve quelques HOWTO consacrés à PHP et à Apache.
- **http://marc.theaimsgroup.com** : site proposant une archive gigantesque de listes consacrées à des outils de l'open source.

Les bibliothèques de scripts

- **http://px.sklar.com** : PX, PHP Code Exchange, la référence en la matière.
- **www.hotscripts.com/PHP/Scripts_and_Programs/index.html** : *Hot Scripts*.
- **http://php.resourceindex.com** : index de ressources PHP.
- **http://freshmeat.net** : le site de référence permettant de trouver tous les logiciels open source possibles et imaginables. Les logiciels sont organisés par catégories (*Communication*, *Database*, *Environnement*, *Langage de programmation*, etc.).

Les modules

- **www.opaque.net/ming** : génération de Flash en PHP.
- **www.pdflib.com** : librairie qui permet de générer des fichiers PDF.
- **www.boutell.com/gd** : librairie qui permet de générer des images.
- **http://mcrypt.sourceforge.net** : librairie qui permet d'accéder à tous les principaux algorithmes de cryptage.

PHP et Windows

- **www.wampserver.com** : téléchargement de Wamp, ainsi que des rubriques forum et FAQ.

- **www.easyphp.org** : site proposant le fantastique logiciel EasyPHP, qui permet de faire fonctionner Apache, PHP, MySQL, phpMyAdmin sous Windows.

Les éditeurs

- **http://notepad-plus.sourceforge.net/fr/site.htm** : excellent éditeur de code fonctionnant sous Windows.
- **www.gnu.org/software/emacs/emacs.html**, **www.gnu.org/software/emacs/windows/ntemacs.html** : l'éditeur par excellence pour Unix/Linux, et qui peut également fonctionner sous Windows.
- **www.emacswiki.org/cgi-bin/emacs-fr**, **www.linux-france.org/article/appli/emacs/index.html** : des sites en français consacrés à Emacs.
- **www.dotemacs.de** : une formidable source d'informations pour personnaliser votre Emacs.

La sécurité

- **http://phpsec.org/projects/vulnerabilities/securityfocus.html** : failles de sécurités de logiciels écrits en PHP.
- **http://blog.php-security.org** : blog consacré à la sécurité autour de PHP.
- **www.hardened-php.net** : projet visant à inclure des extensions de sécurité au sein du projet PHP.

MySQL

- **www.mysql.com** : site officiel du SGBD MySQL. Téléchargement des sources ou des binaires. Documentation récente et téléchargeable dans différents formats.
- **http://phpmyadmin.sourceforge.net** : site désormais officiel du logiciel phpMyAdmin. Possibilité de télécharger la version stable et la version bêta.
- **www.mysqlperformanceblog.com** : blog extrêmement pointu consacré aux optimisations MySQL.

Apache

- **www.apache.org**, **http://httpd.apache.org/docs/2.0/mod/** : site officiel. Téléchargement, documentation, FAQ.

- **www.apacheweek.com** : news et articles réguliers sur Apache et les modules environnants (notamment PHP).

Internet et le Web

Généralités

- **http://fr.wikipedia.org/wiki/Internet** : une histoire d'Internet.
- **http://sunsite.dk/RFC** : un moteur de recherche de RFC (Request for Comments). L'accès à une RFC peut être réalisé par numéros ou par mots-clés. Quelques RFC importantes : HTTP (2616 2617), MAIL (1521 2821 2822).
- **www.w3.org** : le site du World Wide Web Consortium, référence pour tout ce qui touche au monde du Web. Présence de documentations très précises et techniques sur les différentes technologies du Web d'aujourd'hui et de demain.

Les navigateurs

- **www.mozilla.org** : Mozilla est un excellent navigateur gratuit fonctionnant sur une multitude de plateformes.
- **www.mozillazine.org** : site de news concernant Mozilla et les projets environnants.
- **www.mozdev.org** : site consacré à des applicatifs fondés sur le framework Mozilla.

HTML, CSS et Javascript

- **www.htmlhelp.com** : tout ce qu'il faut savoir sur le HTML et les CSS (très bonnes documentations à télécharger).
- **www.htmldog.com/reference/cssproperties** : beaucoup d'informations sur les CSS.
- **http://javascript.internet.com** : de très nombreux exemples.
- **www.w3schools.com/html/html_colors.asp** : site qui permet de voir, en un coup d'œil, les codes couleurs HTML.
- **http://developer.mozilla.org/en/docs/Main_Page** : le site Mozilla consacré aux développeurs. Un must !

Et les blogs…

- www.planet-php.net : blogs consacrés à PHP.
- www.planetmysql.org : blogs consacrés à MySQL.
- www.planetapache.org : blogs consacrés à Apache.
- http://planet.mozilla.org : blogs consacrés au navigateur Firefox.
- http://andigutmans.blogspot.com, www.suraski.net/blog, http://blog.360.yahoo.com/rlerdorf : les blogs des créateurs de PHP.

Divers

- www.ucc.ie/cgi-bin/acronym : un moteur de recherche consacré aux abréviations. Un bon moyen pour savoir ce que veulent dire HTTP, FTP, etc.
- www.dafont.com, www.fontfreak.com, www.1001freefonts.com : sites proposant des fontes gratuites.

19.2. PHP

Les opérateurs

Les opérateurs arithmétiques

Tableau 19.1 : *Opérateurs arithmétiques*

Opérateur	Nom	Résultat
`+`	Addition	Somme de `$x` et `$y`
`–`	Soustraction	Différence de `$x` et `$y`
`*`	Multiplication	Multiplication de `$x` et `$y`
`/`	Division	Division de `$x` et `$y`
`%`	Modulo	Modulo de `$x` et `$y`

Les opérateurs sur les bits

Tableau 19.2 : *Opérateurs sur les bits*

Opérateur	Nom	Résultat
`$x & $y`	ET (AND)	Les bits définis à `1` dans `$x` ET dans `$y` sont placés à `1`
`$x | $y`	OU (OR)	Les bits définis à `1` dans `$x` OU dans `$y` sont placés à `1`
`$x ^ $y`	Xor	Les bits définis à `1` dans `$x` OU dans `$y` sont placés à `1`
`~ $x`	NON (Not)	Les bits qui sont définis à `1` dans `$x` sont placés à `0`, et vice-versa
`$x << $y`	Décalage à gauche	Décale les bits de `$x` dans `$y` par la gauche (chaque décalage équivaut à une multiplication par 2)
`$x >> $y`	Décalage à droite	Décalage des bits de `$x` dans `$y` par la droite (chaque décalage équivaut à une division par 2)

Opérateurs de comparaison

Tableau 19.3 : *Opérateurs de comparaison*

Opérateur	Résultat
`$x == $y`	Vrai si la valeur `$x` est égale à `$y`
`$x === $y`	Vrai si la valeur `$x` est égale à `$y` et si elles sont de même type
`$x != $y`	Vrai si `$x` est différente de `$y`
`$x <> $y`	Vrai si `$x` est différente de `$y`
`$x !== $y`	Vrai si `$x` est différente de `$y` et si elles sont de type différent
`$x > $y`	Vrai si `$x` est supérieure à `$y`
`$x < $y`	Vrai si `$x` est inférieure à `$y`
`$x >= $y`	Vrai si `$x` est supérieure ou égale à `$y`
`$x <= $y`	Vrai si `$x` est inférieure ou égale à `$y`

Opérateurs arithmétiques

<table>
<tr><th colspan="2">Tableau 19.4 : Opérateurs arithmétiques</th></tr>
<tr><th>Opérateur</th><th>Résultat</th></tr>
<tr><td><code>$x and $y</code></td><td>Vrai si <code>$x</code> ET <code>$y</code> sont vraies</td></tr>
<tr><td><code>$x && $y</code></td><td>Vrai si <code>$x</code> ET <code>$y</code> sont vraies (identique à <code>and</code>)</td></tr>
<tr><td><code>$x or $y</code></td><td>Vrai si <code>$x</code> OU <code>$y</code> sont vraies</td></tr>
<tr><td><code>$x || $y</code></td><td>Vrai si <code>$x</code> OU <code>$y</code> sont vraies (identique à <code>or</code>)</td></tr>
<tr><td><code>$x xor $y</code></td><td>Vrai si <code>$x</code> OU <code>$y</code> sont vraies, mais pas les deux</td></tr>
<tr><td><code>! $x</code></td><td>Vrai si <code>$x</code> est fausse</td></tr>
</table>

Priorité des opérateurs

Ce tableau est conçu dans l'ordre croissant de priorité.

Dans l'exemple suivant, l'opérateur == est en dessous de l'opérateur `or` ; il est donc prioritaire :

```
if ($a == b$ or $c)
```

L'expression est donc équivalente à :

```
if (($a == $b) or $c)
```

<table>
<tr><th colspan="2">Tableau 19.5 : Associativité des opérateurs</th></tr>
<tr><th>Associativité</th><th>Opérateurs</th></tr>
<tr><td>Gauche</td><td><code>,</code></td></tr>
<tr><td>Gauche</td><td><code>or</code></td></tr>
<tr><td>Gauche</td><td><code>xor</code></td></tr>
<tr><td>Gauche</td><td><code>and</code></td></tr>
<tr><td>Droite</td><td><code>print</code></td></tr>
<tr><td>Gauche</td><td><code>= += -= *= /= .= %= &= |= ^= ~= <<=>>=</code></td></tr>
<tr><td>Gauche</td><td><code>? :</code></td></tr>
<tr><td>Gauche</td><td><code>||</code></td></tr>
<tr><td>Gauche</td><td><code>&&</code></td></tr>
</table>

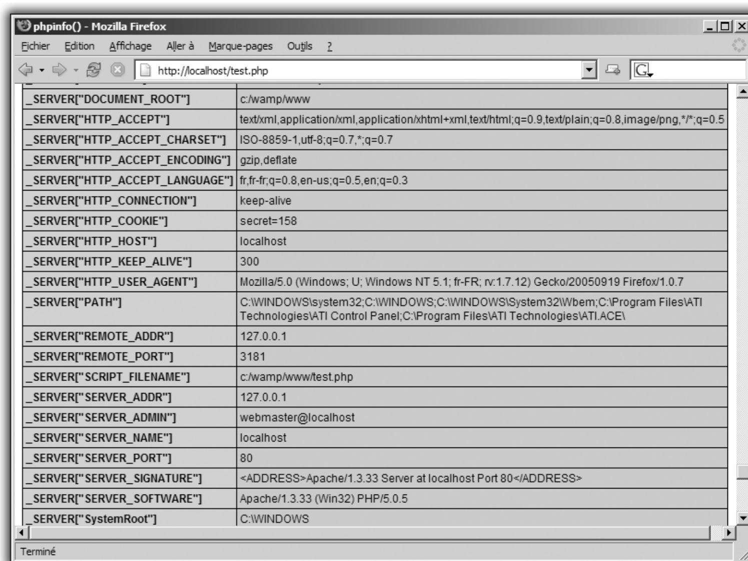
Tableau 19.5 : Associativité des opérateurs	
Associativité	**Opérateurs**
Gauche	|
Gauche	^
Gauche	&
Non associatif	== != ===
Non associatif	< <= > >=
Gauche	<< >>
Gauche	+ - .
Gauche	* / %
Droite	! ~ ++ -- (int) (double) (string) (array) (object) @
Droite	[
Non associatif	new

Les variables prédéfinies

Un script PHP peut accéder à tout moment à un certain nombre de variables prédéfinies. Nous allons lister dans cette partie deux types de variables :

- les variables Apache, qui ne sont disponibles que si vos scripts tournent sur un serveur web Apache ;
- les variables PHP, qui, elles, sont toujours accessibles.

La fonction `phpinfo()` peut être utilisée pour obtenir un aperçu rapide de toutes ces variables. Elle vous permettra aussi de savoir si votre serveur web est Apache.



_SERVER["DOCUMENT_ROOT"]	c:/wamp/www
_SERVER["HTTP_ACCEPT"]	text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
_SERVER["HTTP_ACCEPT_CHARSET"]	ISO-8859-1,utf-8;q=0.7,*;q=0.7
_SERVER["HTTP_ACCEPT_ENCODING"]	gzip,deflate
_SERVER["HTTP_ACCEPT_LANGUAGE"]	fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3
_SERVER["HTTP_CONNECTION"]	keep-alive
_SERVER["HTTP_COOKIE"]	secret=158
_SERVER["HTTP_HOST"]	localhost
_SERVER["HTTP_KEEP_ALIVE"]	300
_SERVER["HTTP_USER_AGENT"]	Mozilla/5.0 (Windows; U; Windows NT 5.1; fr-FR; rv:1.7.12) Gecko/20050919 Firefox/1.0.7
_SERVER["PATH"]	C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\Program Files\ATI Technologies\ATI Control Panel;C:\Program Files\ATI Technologies\ATI.ACE\
_SERVER["REMOTE_ADDR"]	127.0.0.1
_SERVER["REMOTE_PORT"]	3181
_SERVER["SCRIPT_FILENAME"]	c:/wamp/www/test.php
_SERVER["SERVER_ADDR"]	127.0.0.1
_SERVER["SERVER_ADMIN"]	webmaster@localhost
_SERVER["SERVER_NAME"]	localhost
_SERVER["SERVER_PORT"]	80
_SERVER["SERVER_SIGNATURE"]	<ADDRESS>Apache/1.3.33 Server at localhost Port 80</ADDRESS>
_SERVER["SERVER_SOFTWARE"]	Apache/1.3.33 (Win32) PHP/5.0.5
_SERVER["SystemRoot"]	C:\WINDOWS

Figure 19.1 : *Liste de variables proposées par Apache*

Les variables Apache

En fonctionnant avec le serveur web Apache, vous êtes sûr de pouvoir accéder à ces variables. D'autres serveurs web peuvent aussi vous y donner accès. Vous n'avez cependant aucune garantie du pourcentage de variables qui seront disponibles. Il se peut également que certaines d'entre elles portent un nom différent. Quoi qu'il en soit, Apache est, aujourd'hui et de loin, le serveur web le plus utilisé pour faire fonctionner des scripts PHP, et il y a fort à parier que votre hébergeur ait fait ce choix.

$_SERVER['GATEWAY_INTERFACE']

Cette variable contient la version de la spécification CGI utilisée par le serveur web. Par exemple : `CGI/1.1`.

$_SERVER['SERVER_NAME']

Cette variable contient le nom du serveur web sur lequel le script est exécuté. S'il s'agit d'une machine qui fonctionne en *virtual host*

(plusieurs domaines sur la même machine), c'est le nom du *virtual host* qui est retourné. Par exemple : `localhost`.

$_SERVER['SERVER_SOFTWARE']

Cette variable contient une identification technique du serveur web sur lequel est exécuté le script. Par exemple : `Apache/1.3.14 (Unix) (Red–Hat/Linux) mod_perl/1.23`.

Grâce à cette identification, on devine que le serveur web est Apache, que sa version est 1.3.14 et que le système d'exploitation sur lequel il fonctionne est un Linux (distribution Red Hat).

$_SERVER['SERVER_PROTOCOL']

Cette variable contient le nom et la version du protocole qui a été utilisé pour récupérer la page. Par exemple : `HTTP/1.1`.

$_SERVER['REQUEST_METHOD']

Cette variable contient le nom de la méthode qui a été utilisée pour accéder à la page. Les différentes valeurs possibles sont : `GET`, `HEAD`, `POST`, `PUT`.

$_SERVER['QUERY_STRING']

Cette variable contient la *query string* qui a été transmise au script. Si l'URL d'appel est `http://localhost/test.PHP 3?var1 =2&var2=3`, la variable `$_SERVER['QUERY_STRING']` contiendra `var1=2&var2=3`.

$_SERVER['DOCUMENT_ROOT']

Cette variable contient le répertoire racine de votre compte. Cette variable est définie dans le fichier de configuration d'Apache. Il y a de fortes chances pour que cette valeur ne vous serve à rien. Par exemple : `c:/wamp/www`.

$_SERVER['HTTP_ACCEPT']

Cette variable contient la partie `Accept:` de l'en-tête HTTP de la requête courante (si elle existe). Par exemple : `text/xml, application/xml, application/xhtml+xml, text/html;`

q=0.9, image/png, image/jpeg, image/gif;q=0.2, text/plain;q=0.8, text/css, */*;q=0.1.

Certains navigateurs acceptent plus ou moins de formats de fichiers. L'exemple précédent correspond à Netscape. Pour Internet Explorer, la variable contient : image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint, application /vnd.ms-excel, application/msword, */*.

Vous voyez donc qu'Internet Explorer est capable de lire directement des fichiers PowerPoint, Excel ou Word. Cette variable est rarement complète. On sait en effet qu'Internet Explorer accepte le format PNG.

$_SERVER['HTTP_ACCEPT_CHARSET']

Cette variable contient la partie Accept-Charset: de l'en-tête HTTP de la requête courante (si elle existe). Par exemple : ISO-8859-1, utf-8;q=0.66, *;q=0.66.

$_SERVER['HTTP_ACCEPT_ENCODING']

Cette variable contient la partie Accept-Encoding: de l'en-tête HTTP de la requête courante (si elle existe). Par exemple : gzip,deflate,compress,identity.

Cette variable est utilisée par le serveur web pour savoir comment il peut répondre au navigateur. Une fonctionnalité des navigateurs souvent inconnue est de pouvoir recevoir des données compressées et les décompresser à la volée. Le fait qu'il y ait moins de données à transmettre équivaut bien évidemment à une accélération du chargement de la page.

L'exemple suivant envoie de manière compressée une page contenant un poème :

Listing 19-1 : envoi d'un contenu compressé

```
<?php

if (strpos($_SERVER['HTTP_ACCEPT_ENCODING'], "gzip") ===
✂ false)
  die("votre navigateur n'accepte pas les pages
  ✂ compressées");

ob_start();
```

```
?>

<html>

<head>
<title>Le Corbeau et le Renard</title>
</head>

<body>

<pre>
Maître Corbeau, sur un arbre perché,
Tenait en son bec un fromage.
Maître Renard, par l'odeur alléché,
Lui tint à peu près ce langage :
"Hé ! bonjour, Monsieur du Corbeau.
Que vous êtes joli ! que vous me semblez beau !
Sans mentir, si votre ramage
Se rapporte à votre plumage,
Vous êtes le Phénix des hôtes de ces bois. "
A ces mots le Corbeau ne se sent pas de joie ;
Et pour montrer sa belle voix,
Il ouvre un large bec, laisse tomber sa proie.
Le Renard s'en saisit, et dit : "Mon bon Monsieur,
Apprenez que tout flatteur
Vit aux dépens de celui qui l'écoute :
Cette leçon vaut bien un fromage, sans doute. "
Le Corbeau, honteux et confus,
Jura, mais un peu tard, qu'on ne l'y prendrait plus.
</pre>

</body>
</html>

<?php

$contenu = ob_get_contents();

$gzdata  = "\x1f\x8b\x08\x00\x00\x00\x00\x00";
$taille     = strlen($contenu);
$crc       = crc32($contenu);
$gzdata .= gzcompress($contenu, 5);
$gzdata  = substr($gzdata, 0, strlen($gzdata) - 4);
$gzdata .= pack("V",$crc) . pack("V", $taille);

ob_end_clean();

Header('Content-Encoding: gzip');
Header('Vary: Accept-Encoding');
Header('Content-Length: ' . strlen($gzdata));
```

```
echo $gzdata;

?>
```

Si vous commentez la ligne `Header('Content-Encoding: gzip')` (directive HTTP qui indique au navigateur que le contenu est compressé), vous obtenez le résultat suivant :

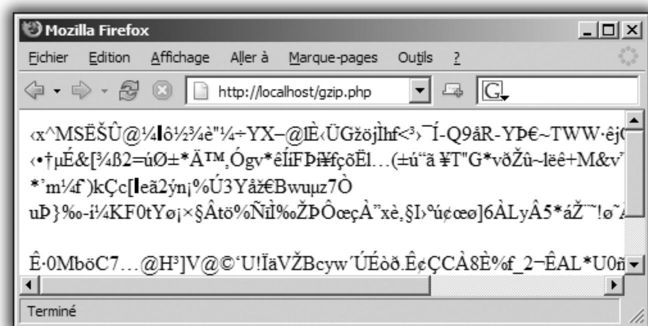


Figure 19.2 : *Sans la fonction header(), le navigateur a l'impression de recevoir du texte brut*

$_SERVER['HTTP_ACCEPT_LANGUAGE']

Cette variable contient la partie `Accept-Language:` de l'en-tête HTTP de la requête courante (si elle existe). Par exemple : `en-us`.

$_SERVER['HTTP_CONNECTION']

Cette variable contient la partie `Connection:` de l'en-tête HTTP de la requête courante (si elle existe). Par exemple : `Keep-Alive`.

$_SERVER['HTTP_HOST']

Cette variable contient la partie `Host:` de l'en-tête HTTP de la requête courante (si elle existe). Par exemple : `localhost`.

$_SERVER['HTTP_REFERER']

Cette variable contient l'adresse de la page qui a mené à la page actuelle. Cette variable n'est pas toujours transmise par le navigateur. Par exemple, supposons qu'une page **http://localhost/liens.html** pointe sur la page **http://localhost/test.php**. Si un internaute passe par *liens.html* pour aller sur *test.php*, le script *test.php* aura alors la possibilité d'accéder à la

variable `$_SERVER['HTTP_REFERER']`, qui contiendra `http://localhost/liens.html`. Cette variable est donc très intéressante lorsque vous voulez savoir par où sont passés les visiteurs pour arriver sur votre page.

$_SERVER['HTTP_USER_AGENT']

Cette variable contient la partie `User_Agent:` de l'en-tête HTTP de la requête courante (si elle existe). Cette variable donne des indications très précises sur le type de navigateur qui a été utilisé pour accéder à la page.

Étudions certaines valeurs possibles et les informations qu'elles donnent.

```
Mozilla/5.0 (Windows; U; Windows NT 5.1;
en-US; rv:1.8) Gecko/20051111 Firefox/1.5
```

Navigateur : Firefox 1.5 en anglais ; système d'exploitation : Windows NT 5.1 (correspondant à Windows XP).

```
Mozilla/4.0 (compatible ; MSIE 5.5 ; Windows NT 5.0)
```

Navigateur : Internet Explorer 5.5 ; système d'exploitation : Microsoft Windows NT 5.0.

```
Mozilla/4.0 (compatible ; MSIE 6.0 ; Windows NT 5.0 ;
KITV4.7 Wanadoo ; .NET CLR 1.0.2914)
```

Navigateur : Internet Explorer 6.0 ; système d'exploitation : Microsoft Windows NT 5.0 ; fournisseur d'accès : Wanadoo.

```
Mozilla/4.75 [fr] (Win98 ; U)
```

Navigateur : Netscape 4.75 ; système d'exploitation : Microsoft Windows 98.

```
Mozilla/4.5 [fr] (Macintosh ; I ; PPC)
```

Navigateur : Netscape 4.5 ; Type de machine : Macintosh Power PC.

```
Mozilla/3.01-C-MACOS8 (Macintosh ; I ; PPC)
```

Navigateur : Netscape 3.01 ; système d'exploitation : Mac OS 8 ; type de machine : Macintosh Power PC.

```
Mozilla/4.77 [en] (X11 ; U ; Linux 2.0.38 i386)
```

Navigateur : Netscape 4.77 ; système d'exploitation : Linux ; noyau (kernel) : 2.0.38 ; type de machine : PC avec un processeur au moins compatible 386.

```
Mozilla/5.0 (X11 ; U ; Linux i386 ; en-US ; rv:0.9.4)
✂ Gecko/20011019 Netscape6/6.2
```

Navigateur : Netscape 6.2 (compatible Netscape 5.0) ; moteur de rendu HTML : Gecko compilé le 19 octobre 2001 ; système d'exploitation : Linux ; système de fenêtrage : X11 ; type de machine : PC avec un processeur au moins compatible 386.

```
Mozilla/4.0 (compatible ; MSIE 5.0 ; Linux 2.2.16 i586)
Opera 5.0 [en]
```

Navigateur : Opera ; système d'exploitation : Linux ; noyau (kernel) : 2.2.16 ; type de machine : PC avec un processeur Pentium (au moins 1).

```
Mozilla/4.7 [en] (X11; U; SunOS 5.7 sun4u)
```

Navigateur : Netscape 4.7 ; système d'exploitation : SunOS ; système de fenêtrage : X11 ; type de machine : machine de la marque Sun.

Cette variable, couplée à un petit script utilisant les `regexp`, peut donc être très intéressante pour obtenir des statistiques sur les systèmes des internautes qui viennent sur votre page. Ces statistiques pourront être utilisées pour savoir avec quel navigateur votre site doit être compatible en priorité (parions toutefois qu'il s'agira d'Internet Explorer 4 sous PC).

PHP fournit aussi la fonction `get_browser()` pour donner des informations sur l'internaute à partir d'une chaîne de type `$_SERVER['HTTP_USER_AGENT']`. Cette fonction retourne un objet contenant un certain nombre de propriétés :

- le nom du navigateur ;
- la plateforme ;
- le comportement du navigateur (gère-t-il les frames, les cookies, etc. ?).

```
<?php

function prop_navig ($tab)
{
  while (list ($clef, $valeur) = each ($tab))
  {
    $str .= "<b>$clef:</b> $valeur<br>\n";
  }
```

```
    return $str;
  }

  echo $_SERVER['HTTP_USER_AGENT']."<hr/>\n";
  $navigateur = get_browser();
  echo prop_navig ((array) $navigateur);

  return 0;

  ?>
```

Cette fonction nécessite toutefois de récupérer un fichier (`browscap.ini`) et de changer un paramètre de configuration dans *php.ini* (`browscap = "C:/wamp/browscap.ini"`). La base de données *browscap.ini* n'étant plus réellement mise à jour, cette fonction a perdu une grande partie de son intérêt. Le fichier *browscap.ini* peut être récupéré sur le site **www.garykeith.com/browsers/downloads.asp**.

$_SERVER['REMOTE_ADDR']

Cette variable contient l'adresse IP de l'internaute qui a demandé la page.

Pour des applicatifs sensibles (paiement sécurisé, zone d'administration), il peut être intéressant de logger cette information (ainsi que l'heure) dans une base de données ou un fichier. En cas d'intrusion, cette IP peut vous permettre de remonter jusqu'au fautif. Les fournisseurs d'accès sont en effet dans l'obligation de conserver les logs de connexion de leurs abonnés afin de pouvoir les identifier à partir d'une IP et d'une heure.

$_SERVER['REMOTE_PORT']

Cette variable contient le port qu'utilise la machine cliente pour se connecter au serveur web. Dans le protocole TCP/IP, le client et le serveur communiquent par port. C'est de cette manière qu'une machine est capable de savoir que la requête qu'elle vient de recevoir est une requête HTTP, FTP, etc. Par exemple : `1242`.

$_SERVER['SCRIPT_FILENAME']

Cette variable contient le chemin absolu du script actuellement exécuté. Par exemple : `c:/wamp/www/test.php`.

$_SERVER['SERVER_ADMIN']

Cette variable contient l'adresse de courriel du responsable du site. Cette variable n'est initialisée que si les fichiers de configuration d'Apache sont complets. Par exemple : `fx@micro.com`.

$_SERVER['SERVER_PORT']

Cette variable contient le numéro de port utilisé pour communiquer avec le serveur. Le port du protocole HTTP est 80. Il est cependant possible de demander au serveur web d'écouter sur un autre port (par exemple 8080). L'URL pour accéder à ce site sera alors : `http://localhost:8080`. Si 80 est le port par défaut, vous pouvez toujours ajouter `80` à la fin d'une URL : `http://www.google.fr:80`.

L'avantage de pouvoir jouer sur les ports est de permettre d'avoir plusieurs sites web hébergés sur la même IP et sur le même nom de domaine. Globalement, si l'IP correspond à l'adresse du bâtiment (`12, rue carnot`) et le nom de domaine à sa dénomination (`bureau de poste paris 15`), le port pourrait permettre d'accéder à un service (`service des contentieux`).

$_SERVER['SCRIPT_NAME']

Cette variable contient le chemin relatif du script en cours. Par exemple, si vous tapez l'URL `http://localhost/infos.php?param=b`, cette variable contient `/infos.php`.

$_SERVER['REQUEST_URI']

Cette variable contient l'URL (relative) qui a été utilisée pour appeler cette page. Par exemple, si vous tapez l'URL `http://localhost/infos.php?param=b`, cette variable contient `/infos.php?param=b`.

Les variables PHP

$_SERVER['argv']

Cette variable contient un tableau des arguments qui ont été passés au script si celui-ci a été exécuté en ligne de commande. Si le script a été appelé avec la méthode `GET`, cette variable contient la *Query String*.

$_SERVER['argc']

Cette variable contient le nombre d'arguments passés sur la ligne de commande.

$_SERVER['PHP_SELF']

Cette variable contient le chemin relatif du script en cours. Par exemple, si vous appelez l'URL **http://localhost/datas/index.php**, alors `$_SERVER['PHP_SELF']` contient `/datas/index.php`.

$_COOKIE

Cette variable contient un tableau associatif des variables passées au script par les cookies.

$_GET

Cette variable contient un tableau associatif des variables passées au script par la méthode `GET`.

$_POST

Cette variable contient un tableau associatif des variables passées au script par la méthode `POST`.

$_FILES

Cette variable contient un tableau associatif des fichiers qui ont été envoyés sur le serveur par la méthode `POST`.

Si, dans votre formulaire, le champ `file` porte le nom `userfile`, les propriétés suivantes deviennent disponibles…

- `$_FILES['userfile']['name']` : contient le nom du fichier que vous avez sélectionné sur votre disque.
- `$_FILES['userfile']['type']` : contient le type du fichier (par exemple `image/gif`).
- `$_FILES['userfile']['size']` : contient la taille du fichier en bytes.
- `$_FILES['userfile']['tmp_name']` : contient le nom temporaire du fichier au moment du transfert.

- $_FILES['userfile']['error'] : contient un code d'erreur (0 si tout s'est bien déroulé).

$_ENV

Cette variable contient un tableau associatif des variables d'environnement accessibles par un script PHP.

$HTTP_RAW_POST_DATA

Cette variable contient le corps d'une requête transmise en mode POST.



Figure 19.3 : *Affichage de $HTTP_RAW_POST_DATA suite à la soumission du formulaire de DVD.*



Directive de ***php.ini***

Cette variable n'est initialisée que lorsque la directive de configuration always_populate_raw_post_data est à on dans le fichier php.ini. Veillez également à vérifier que le formulaire est bien envoyé en mode POST.

Les mots réservés

PHP contient un certain nombre de mots réservés, qui ne doivent pas être utilisés pour créer des variables, des fonctions ou des constantes.

Mots réservés de PHP				
and	or	xor	__FILE__	exception (PHP5)
__LINE__	array()	as	break	case
class	const	continue	declare	default

Mots réservés de PHP				
die()	do	echo()	else	elseif
empty()	enddeclare	endfor	endforeach	endif
endswitch	endwhile	eval()	exit()	extends
for	foreach	function	global	if
include()	include _once()	isset()	list()	new
print()	require()	require _once()	return()	static
switch	unset()	use	var	while
__FUNCTION__	__CLASS__	__METHOD__	final (PHP5)	php_user _filter (PHP5)
interface (PHP5)	implements (PHP5)	extends	public (PHP5)	private (PHP5)
protected (PHP5)	abstract (PHP5)	clone(PHP5)	try(PHP5)	catch (PHP5)
throw (PHP5)	cfunction (PHP4 uniquement)	old _function (PHP4 uniquement)	this (PHP5 uniquement)	

Les différences entre PHP 3 et PHP 4

Aujourd'hui, PHP 4 est la version de PHP le plus aboutie et le plus stable. Les gains en termes de fonctionnalités et de rapidité sont absolument énormes.

Il faut noter que certains hébergeurs continuent à proposer la version 3. Il est donc intéressant de connaître les différences entre ces deux versions. En effet, si vous ne connaissez pas la version de l'interpréteur qui sera utilisé pour faire fonctionner votre script, il vaut mieux écrire un code compatible PHP 3 et 4. Au contraire, si vous êtes sûr qu'il s'agit de la version 4, vous pouvez alors profiter au maximum des nouvelles fonctionnalités et optimiser votre code en ce sens.

Le parseur

Avec PHP 4, un script est d'abord intégralement parsé avant d'être exécuté. Un script ne doit contenir aucune erreur de syntaxe pour pouvoir être exécuté.

De plus, il est désormais impossible d'avoir une structure de contrôle qui s'étend sur plusieurs fichiers (le `case` dans un fichier et les `switch` dans un autre). Il est, par contre, tout à fait possible d'inclure (`include()`) d'autres fichiers au sein d'une structure de contrôle.

L'affichage d'erreur

PHP 4 est beaucoup moins permissif en ce qui concerne les erreurs de syntaxe. Il ne faut donc pas être étonné qu'un code apparemment « sain » avec PHP 3 génère des avertissements avec PHP 4.

L'initialisation des variables

Alors qu'il était possible, en PHP 3, d'initialiser un attribut avec n'importe quel type de données, PHP 4 n'autorise, quant à lui, que des valeurs statiques :

```
class test
{
  var maintenant = time();
  // valide en PHP 3 mais pas en PHP 4
  var pays = "france";
  // valide en PHP 3 et 4
}
```

Il est donc préférable d'utiliser les constructeurs pour initialiser ces attributs.

Comportement des fonctions empty() et isset()

Une variable qui ne contient que le caractère `0` est désormais considérée comme vide :

```
$str = "0";

if (empty($str)) echo "OK";
else echo "KO";
// PHP 3  affiche KO
```

```
// PHP 4 affiche OK
```

Une variable contenant la valeur "" vide est considérée comme initialisée avec PHP 4, mais pas avec PHP 3 :

```
$str = "";

if (isset($str)) echo "OK";
else echo "KO";
// PHP 3  affiche KO
// PHP 4 affiche OK
```

Substitution de variables complexes dans les chaînes de caractères

Il est désormais possible, avec PHP 4, de substituer, au sein de chaînes de caractères, des éléments complexes : attribut d'une classe, élément d'un tableau multidimensionnel.

Version PHP 3 (obligatoire) :

```
$str =  "monsieur " . $personne->nom
. " possède " . $compte["pea"]["nb"] . "actions";
```

Version PHP 4 (possible) :

```
$str =  "monsieur $personne->nom possède
        {$compte["pea"]["nb"]} actions";
```

Comme vous pouvez le constater, la notation entre parenthèses est utilisée pour l'élément du tableau `$compte`.

Les extensions

Les extensions écrites pour PHP 3 ne fonctionneront pas avec PHP 4. Quand vous téléchargez une extension sur le Net, veillez donc à récupérer la version qui correspond à votre système.

La création des cookies

PHP 3 créait les cookies dans l'ordre inverse de leur initialisation (`setcookie()`) dans le code. Ce n'est plus le cas avec PHP 4.

Ce qu'apporte PHP 4

- Plus de rapidité, moins de consommation de mémoire ;
- l'affichage tampon (*ouput buffering*) ;
- la comparaison de type ;
- la syntaxe dite du *here printing* ;
- la gestion des sessions ;
- le support du protocole FTP ;
- le fonctionnement par comptage de référence (*reference counting*) ;
- l'assignation de variables par référence (deux variables pointent sur la même donnée, la modification de l'une entraînant la modification de l'autre) : `$var1 &= $var2;` (`$var2` n'est qu'un alias de `$var1`).

Les différences entre PHP 4 et PHP 5

Les nouveautés

Les apports de PHP 5 sont nombreux :

- la présence d'un SGBD au sein même de l'environnement (SQLite) ;
- l'extension PDO qui permet d'utiliser les mêmes fonctions pour accéder à tout type de base de données ;
- des technologies majeures dans le domaine de l'entreprise (SOAP, XML) ;
- une dimension objet plus aboutie.

Les incompatibilités

La fonction `array_merge()` n'autorise désormais que les tableaux en paramètres.

19.3. MySQL

Les types

Tableau 19.6 : *Les types de MySQL*		
Nom	**Définition**	**Espace**
`TINYINT`	Entier compris entre -128 et 127 (ou entre 0 et 255 si `UNSIGNED`)	1 octet
`SMALLINT`	Entier compris entre -32 768 et 32767 (ou entre 0 et 65 535 si `UNSIGNED`)	2 octets
`MEDIUMINT`	Entier compris entre -8 388 608 et 8 388 607 (ou entre 0 et 16 777 215 si `UNSIGNED`)	3 octets
`INT`	Entier compris entre -2 147 483 648 et 2 147 483 647 (ou entre 0 et 4 294 967 295 si `UNSIGNED`)	4 octets
`INTEGER`	Synonyme de `INT`	4 octets
`BIGINT`	Entier compris entre -9 223 372 036 854 775 808 et 9 223 372 036 854 775 807 (ou entre 0 et 18 446 744 073 709 551 615 si `UNSIGNED`)	8 octets
`FLOAT(M,D)`	Nombre à virgule (`M` correspond à la taille d'affichage et `D` au nombre de décimales)	4 octets
`DOUBLE(M,D)`	Nombre à virgule (double précision)	8 octets
`REAL(M,D)`	Synonyme de `DOUBLE`	8 octets
`DECIMAL(M,D)`	Permet de stocker un nombre à virgule dans une chaîne de caractères	`M` octets
`NUMERIC(M,D)`	Synonyme de `DECIMAL`	`M` octets
`DATE`	Date comprise entre 1000-01-01 et 9999-12-31	3 octets

Tableau 19.6 : Les types de MySQL		
Nom	**Définition**	**Espace**
DATE	Date et heure comprises entre 1000-01-01 00:00:00 et 9999-12-31 23:59:59	8 octets
TIMESTAMP	Un *timestamp* compris entre 1970-01-01 00:00:00 et une date en 2037	4 octets
TIME	Une heure comprise entre -838:59:59 et 838:59:59	3 octets
YEAR	Année comprise entre 1901 et 2155	1 octet
CHAR	Permet de stocker de 1 à 255 caractères	
VARCHAR / TINYBLOB / TINYTEXT	Permet de stocker de 1 à 255 caractères	
BLOB / TEXT	Permet de stocker au maximum 65 535 caractères	
MEDIUMBLOB / MEDIUMTEXT	Permet de stocker au maximum 16 777 215 caractères	
LONGBLOB / LONGTEXT	Permet de stocker au maximum 4 294 967 295 caractères	

Les fonctions

MySQL dispose d'un certain nombre de fonctions qui peuvent être appelées au sein d'une requête.

Cette ligne permet de calculer `5` modulo `2` et de placer le résultat dans `n` :

```
SELECT mod(5,2) AS n;
```

Cette ligne permet d'obtenir un nombre aléatoire :

```
SELECT RAND();
```

Cette ligne permet de lister les élèves nés en février :

```
SELECT * FROM eleve WHERE MONTH(date) = '2'
```

Ce code permet de mettre à jour la colonne *nomprenom* de la table *matable* en regroupant pour chaque ligne le nom et le prenom séparé par le caractère – :

```
UPDATE matable SET nomprenom = CONCAT(nom,"-",prenom)
```

Avant la requête, la table contient :

Tableau 19.7 : *Table avant la requête*		
nomprenom	**nom**	**prenom**
	jean	dutour
	paul	bernager

Après la requête, elle contient :

Tableau 19.8 : *Table après la requête*		
nomprenom	**nom**	**prenom**
jean-dutour	jean	dutour
paul-beranger	paul	beranger

Chaque paramètre d'une fonction peut être une valeur fixe ou le nom d'une colonne :

```
UPDATE matable SET moyenneannee = (moyenne1 + moyenne2 +
✂ moyenne3) / 3;
```

Cette requête permet de calculer la moyenne annuelle de tous les élèves.

Les fonctions mathématiques

Tableau 19.9 : *Fonctions mathématiques de MySQL*	
Nom	**Fonction**
`ABS(X)`	Retourne la valeur absolue de `X`
`SIGN(X)`	Retourne `-1`, `0` ou `1` selon que `X` est négatif, nul ou positif
`MOD(N,M)`	Retourne le reste de la division de `N` avec `M` (l'opérateur modulo `%` peut aussi être utilisé)
`FLOOR(X)`	Retourne l'entier inférieur ou égal à `X`

Tableau 19.9 : *Fonctions mathématiques de MySQL*	
Nom	**Fonction**
`CEILING(X)`	Retourne l'entier supérieur ou égal à `X`
`ROUND(X)`	Retourne l'entier le plus proche
`EXP(X)`	Retourne l'exponentielle de `X`
`LOG(X)`	Retourne le logarithme de `X`
`LOG10(X)`	Retourne le logarithme décimal de `X`
`POW(X,Y)`	Retourne `X` à la puissance `Y`
`SQRT(X)`	Retourne la racine carrée de `X`
`PI()`	Retourne la valeur de `PI`
`COS(X)`	Retourne le cosinus de `X` (en radians), les autres fonctions trigonométriques sont également disponibles
`RAND()`	Retourne une valeur aléatoire
`MIN(X,Y,...)`	Retourne la plus petite valeur de la liste
`MAX(X,Y,...)`	Retourne la plus grande valeur de la liste
`DEGREES(X)`	Permet de convertir des radians en degrés
`RADIANS(X)`	Permet de convertir des degrés en radians
`TRUNCATE(X,D)`	Permet de ne conserver que `D` décimales de `X`

Les fonctions de manipulation de chaînes

Tableau 19.10 : *Fonctions de manipulation de chaînes de MySQL*	
Nom	**Fonction**
`ASCII(str)`	Retourne le code ASCII du premier caractère de la chaîne
`CONV(N,from_base, to_base)`	Permet de convertir un nombre d'une base dans une autre
`CHAR(N,...)`	Permet de construire une chaîne à partir de codes ASCII
`CONCAT(str1,str2, ...)`	Permet de concaténer plusieurs chaînes

***Tableau 19.10 :** Fonctions de manipulation de chaînes de MySQL*	
Nom	**Fonction**
CONCAT_WS (separator, str1, str2,...)	Permet de concaténer plusieurs chaînes en les joignant avec la chaîne separator
LENGTH(str)	Retourne la longueur de la chaîne
LOCATE(substr,str)	Retourne la position de la première occurrence de la chaîne substr dans la chaîne str
INSTR(str,substr)	Synonyme de LOCATE
LPAD(str,len, padstr)	Permet d'augmenter la longueur de str et de combler l'espace vide (à gauche) avec la chaîne padstr
RPAD(str,len, padstr)	Permet d'augmenter la longueur de str et de combler l'espace vide (à droite) avec la chaîne padstr
LEFT(str,len)	Permet de récupérer la sous-partie gauche de str (la sous-partie contient len caractères)
RIGHT(str,len)	Permet de récupérer la sous-partie droite de str (la sous-partie contient len caractères)
SUBSTRING(str,pos, len)	Permet d'extraire une sous-chaîne de str (la sous-chaîne fait len caractères et commence à la position pos)
LTRIM(str)	Permet de supprimer les caractères blancs à gauche de str
RTRIM(str)	Permet de supprimer les caractères blancs à droite de str
TRIM(str)	Permet de supprimer les caractères blancs aux extrémités de str
SPACE(N)	Retourne une chaîne contenant N espaces
REPLACE(str,from _str,to_str)	Permet de remplacer une chaîne dans une autre
REPEAT(str,count)	Permet de répéter une même chaîne count fois
REVERSE(str)	Permet d'inverser l'ordre des caractères
INSERT(str,pos,len, newstr)	Permet d'insérer une chaîne dans une autre
ELT(N,str1,str2, str3,...)	Retourne le énième élément de la liste

Tableau 19.10 : Fonctions de manipulation de chaînes de MySQL	
Nom	**Fonction**
`FIELD(str,str1, str2,str3,...)`	Retourne la position de `str` sur la liste composée par `(str1, str2, str3,...)`
`LOWER(str)`	Retourne la chaîne `str` en minuscules
`UPPER(str)`	Retourne la chaîne `str` en majuscules
`LOAD_FILE(file_name)`	Permet de lire le contenu d'un fichier et de le retourner en tant que chaîne

Les fonctions de manipulation de dates

Tableau 19.11 : Fonctions de manipulation de dates de MySQL	
Nom	**Fonction**
`DAYOFWEEK(date)`	Retourne le jour de la semaine (`0` correspond à dimanche)
`WEEKDAY(date)`	Retourne le jour de la semaine (`6` correspond à dimanche)
`DAYOFMONTH(date)`	Retourne le jour du mois
`DAYOFYEAR(date)`	Retourne le jour de l'année
`MONTH(date)`	Retourne le mois
`DAYNAME(date)`	Retourne le nom du jour
`MONTHNAME(date)`	Retourne le nom du mois
`QUARTER(date)`	Retourne le trimestre
`WEEK(date)`	Retourne la semaine
`YEAR(date)`	Retourne l'année
`YEARWEEK(date)`	Retourne le jour dans l'année
`HOUR(time)`	Retourne l'heure
`MINUTE(time)`	Retourne la minute
`SECOND(time)`	Retourne la seconde
`TO_DAYS(date)`	Convertit une date en nombre de jours
`FROM_DAYS(N)`	Convertit un nombre de jours en date

Tableau 19.11 : *Fonctions de manipulation de dates de MySQL*	
Nom	**Fonction**
`CURDATE()`	Retourne la date actuelle
`CURTIME()`	Retourne l'heure actuelle
`NOW()`	Retourne la date et l'heure actuelles
`SEC_TO_TIME(time)`	Convertit des secondes en heures
`TIME_TO_SEC(time)`	Convertit une heure en secondes

19.4. Les caractères HTML spéciaux

Tableau 19.12 : *Caractères HTML spéciaux*			
Résultat	**Description (anglaise)**	**Code HTML**	**Code caractère**
"	Guillemet	`"`	`"`
&	Et commercial (esperluette)	`&`	`&`
<	Inférieur à	`<`	`<`
>	Supérieur à	`>`	`>`
	Espace non sécable	` `	` `
¡	Point d'exclamation inversé	`¡`	`¡`
¤	Devise	`¤`	`¤`
¢	Centime	`¢`	`¢`
£	Livre	`£`	`£`
¥	Yen	`¥`	`¥`
¦	Barre verticale brisée	`&bvbar;`	`¦`
§	Section	`§`	`§`
¨	Tréma (dieresis)	`¨`	`¨`
©	Copyright	`©`	`©`
ª	Indicateur ordinal féminin	`ª`	`ª`

Tableau 19.12 : *Caractères HTML spéciaux*			
Résultat	**Description (anglaise)**	**Code HTML**	**Code caractère**
«	Chevron gauche	«	«
¬	Négation	¬	¬
-	Trait d'union		
®	Marque déposée	®	®
¯	Accent macron	¯	¯
°	Degré	°	°
±	Plus ou moins	±	±
^{2}	Puissance carrée	²	²
^{3}	Puissance cubique	³	³
´	Accent aigu	´	´
µ	Micro	µ	µ
¶	Paragraphe	¶	¶
·	Point centré	·	·
¸	Cédille	¸	¸
^{1}	Exposant 1	¹	¹
º	Indicateur ordinal masculin	º	º
»	Chevron droit	»	»
¼	Fraction 1/4	¼	¼
½	Fraction 1/2	½	½
¾	Fraction 3/4	¾	¾
¿	Point d'interrogation inversé	¿	¿
×	Multiplication	×	×
÷	Division	÷	÷
À	A accent grave	À	À
Á	A accent aigu	Á	Á
Â	A accent circonflexe	Â	Â

Tableau 19.12 : *Caractères HTML spéciaux*

Résultat	Description (anglaise)	Code HTML	Code caractère
Ã	A tilde	Ã	Ã
Ä	A tréma	Ä	Ä
Æ	AE collé	Æ	Æ
Ç	C cédille majuscule	Ç	Ç
È	E accent grave	È	È
É	E accent aigu	É	É
Ê	E accent circonflexe	Ê	Ê
Ë	E tréma	Ë	Ë
Ø	O barré	Ø	Ø
Ù	U accent grave	Ù	Ù
Ú	U accent aigu	Ú	Ú
Û	U accent circonflexe	Û	Û
Ü	U tréma	Ü	Ü
ß	Beta	ß	ß
à	a accent grave	à	à
á	a accent aigu	á	á
â	a accent circonflexe	â	â
ã	a tilde	ã	ã
ä	a tréma	ä	ä
a	a rond en chef	å	å
æ	ae collé	æ	æ
ç	c cédille	ç	ç
è	e accent grave	è	è
é	e accent aigu	é	é
ê	e accent circonflexe	ê	ê
ë	e tréma	ë	ë
î	i accent circonflexe	î	î

***Tableau 19.12 :** Caractères HTML spéciaux*			
Résultat	**Description (anglaise)**	**Code HTML**	**Code caractère**
ï	i tréma	ï	ï
ñ	n tilde	ñ	ñ
ô	o accent circonflexe	ô	ô
ö	o tréma	ö	ö
ø	o barré	ø	ø
ù	u accent grave	ù	ù
ú	u accent aigu	ú	ú
û	u accent circonflexe	û	û
ü	u tréma	ü	ü
ÿ	y tréma	ÿ	ÿ
Œ	OE collé	Œ	Œ
œ	oe collé	œ	œ
ˆ	Accent circonflexe	ˆ	ˆ
˜	Tilde	˜	˜
–	Tiret demi-cadratin	–	–
—	Tiret cadratin	—	—
‘	Apostrophe à gauche	‘	‘
’	Apostrophe à droite	’	’
‚	Virgule	‚	‚
“	Guillemets à gauche	“	“
”	Guillemets à droite	”	”
„	Double virgule inférieure	„	„
‰	Pour mille	‰	‰
‹	Guillemet simple gauche	‹	‹
›	Guillemet simple droit	›	›
€	Euro	€	€
™	Marque commerciale	™	™

19.5. Les feuilles de styles : CSS

Les unités de longueur

Il existe deux types d'unités de longueur relatives et absolues.

Les unités relatives

- `em` : ems, la hauteur d'un élément de la fonte.
- `ex` : x-height, la hauteur de la lettre *x*.
- `px` : pixels.

Les unités absolues

- `in` : inches/pouces (1 in = 2,54 cm).
- `cm` : centimètres (1 cm = 10 mm).
- `mm` : millimètres.
- `pt` : points (1 pt = 1/72 in).
- `pc` : picas (1 pc = 12 pt).



Notation des unités

Le nombre doit précéder l'unité, et sans espace.

- `12 cm` : non valide.
- `12px` : valide.

Les unités de couleur

Les couleurs peuvent être exprimées de différentes manières.

- Format 1 : `#rrggbb`, où `rr`, `gg` et `bb` sont des valeurs hexadécimales (c'est-à-dire `#00cc00`).
- Format 2 : `#rgb`, où `r`, `g` et `b` sont des valeurs hexadécimales (c'est-à-dire `#0c0`).
- Format 3 : `rgb(x,y,z)` où `x`, `y` et `z` sont des valeurs décimales comprises entre 0 et 255 (c'est-à-dire `rgb(0,204,0)`).
- Format 4 : `rgb(y%,y%,y%)`, où `x`, `y` et `z` sont des valeurs comprises entre 0.0 et 100.0 inclus (c'est-à-dire `rgb(0%,80%,0%)`).

Les URL

Pour spécifier des fonds ou des puces, il est possible de faire appel à des images. Voici différentes manières de spécifier l'image *fond.gif* :

```
BODY { background: url(fond.gif); }

BODY { background: url(http://localhost/images/fond
.gif); }

BODY { background: url("fond.gif");}

BODY { background: url('../fond.gif'); }
```

Les propriétés

Les propriétés des fontes

Tableau 19.13 : *Propriétés des fontes*	
Nom de la propriété	**Valeurs possibles**
font-family	i.e. : "New Century Schoolbook", Times, serif | Helvetica
font-style	normal | italic | oblique
font-variant	normal | small-caps
font-weight	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900
font-size	xx-small | x-small | small | medium | large | x-large | xx-large | larger | smaller | <num>

```
<html>

<style type="text/css">
<!--
.test1
{ font-family:verdana, sans-serif; font-style:italic;
  font-size:12px;}
.test2
{ font-family: "New Century Schoolbook", Times, serif ;
font-size: 150%; font-weight: bold;
font-variant: small-caps; }
```

```
-->
</style>

<body>

<table width="95%" border="1">

<tr><td class="test1">
Maître Corbeau, sur un arbre perché, <br/>
Tenait en son bec un fromage.

</td></tr>

<tr><td class="test2">
Maître Corbeau, sur un arbre perché, <br/>
Tenait en son bec un fromage.

</td></tr>

</table>

</body>
</html>
```



Figure 19.4 : *Affichage dans un navigateur*

Les propriétés des couleurs et des fonds

Tableau 19.14 : *Les propriétés des couleurs et des fonds*	
Nom de la propriété	**Valeurs possibles**
color	<couleur>
background-color	<couleur>
background-image	<url>
background-repeat	repeat | repeat-x | repeat-y | no-repeat
background -attachment	scroll | fixed

Tableau 19.14 : *Les propriétés des couleurs et des fonds*	
Nom de la propriété	**Valeurs possibles**
background-position	top | center | bottom | left | center | right

```
<html>

<style type="text/css">
<!--
.test1
{ color: blue; background-repeat: no-repeat;
background-position: right;  background-image:
   url(http://www.google.com/intl/en/images/logo.gif); }
.test2
{ color: #000000;  background-image: url(bg.jpg);
background-image:
   url(http://www.google.com/intl/en/images/logo.gif); }
-->
</style>

<body>

<table width="95%" border="1">

<tr><td class="test1">
Maître Corbeau, sur un arbre perché, <br />
Tenait en son bec un fromage.

</td></tr>

<tr><td class="test2">
Maître Corbeau, sur un arbre perché, <br />
Tenait en son bec un fromage.

</td></tr>

</table>

</body>
</html>
```

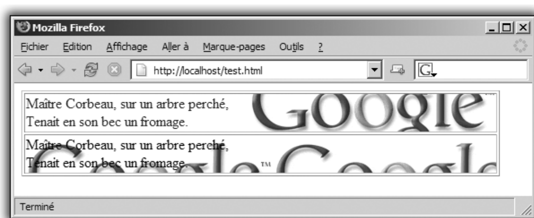


Figure 19.5 : *Exemple d'affichage d'images sur des fonds*

Les propriétés des textes

Tableau 19.15 : *Les propriétés des textes*

Nom de la propriété	Valeurs possibles
word-spacing	<num>
letter-spacing	<num>
text-decoration	underline | overline | line-through | blink
vertical-align	baseline | sub | super | top | text-top | middle | bottom | text-bottom
text-transform	none | capitalize | uppercase | lowercase
text-alignment	left | right | center | justify
text-indent	<num>
line-height	<num>

```
<html>

<style type="text/css">
<!--
.test1
{ word-spacing : 16px; text-decoration: overline;
  text-transform: uppercase; text-align: right;
  line-height: 200%; }
.test2 { letter-spacing : 4px; text-indent: 2em; }
-->
</style>

<body>

<table width="95%" border="1">

<tr><td class="test1">
Maître Corbeau, sur un arbre perché, <br/>
Tenait en son bec un fromage.

</td></tr>

<tr><td class="test2">
Maître Corbeau, sur un arbre perché, <br/>
Tenait en son bec un fromage.

</td></tr>
```

```
</table>

</body>
</html>
```



Figure 19.6 : *Affichage dans un navigateur*

Les propriétés des boîtes

Tableau 19.16 : *Les propriétés des boîtes*

Nom de la propriété	Valeurs possibles
margin-top	<num>
margin-right	<num>
margin-bottom	<num>
margin-left	<num>
margin	<num>
padding-top	<num>
padding-right	<num>
padding-bottom	<num>
padding-left	<num>
padding	<num>
border-top-width	thin | medium | thick | <num>
border-right-width	thin | medium | thick | <num>
border-bottom-width	thin | medium | thick | <num>
border-left-width	thin | medium | thick | <num>

Tableau 19.16 : *Les propriétés des boîtes*	
Nom de la propriété	**Valeurs possibles**
border-width	thin	medium	thick	<num>
border-color	<couleur>
border-style	none	dotted	dashed	solid	double	groove	ridge	inset	outset
border-top	<border-top-width>	<border-style>	<couleur>
border-right	<border-top-width>	<border-style>	<couleur>
border-bottom	<border-top-width>	<border-style>	<couleur>
border-left	<border-top-width>	<border-style>	<couleur>
border	<border-top-width>	<border-style>	<couleur>
width	<num>
height	<num>
float	left	right	none
clear	none	left	right	both

```
<html>

<style type="text/css">
<!--
.test1
{ border-style : dotted; border-top-width : thick;
  padding-bottom: 20px; padding-left: 30px}
.test2
{ padding : 2em; border-color: #800080 ; float: right; }
-->
</style>

<body>

<table width="95%" class="test1">

<tr><td>
Maître Corbeau, sur un arbre perché, <br/>
Tenait en son bec un fromage.
</td></tr>
```

```
</table>

<br/>

<table width="95%" border=1 class="test2">

<tr><td>
Maître Corbeau, sur un arbre perché, <br/>
Tenait en son bec un fromage.
</td></tr>

</table>

</body>
</html>
```



Figure 19.7 : *Affichage du résultat*

Les propriétés des listes

Tableau 19.17 : *Les propriétés des listes*	
Nom de la propriété	**Valeurs possibles**
display	block | inline | list-item | none
white-space	normal | pre | nowrap
list-style-type	disc | circle | square | decimal | lower-roman | upper-roman | lower-alpha | upper-alpha | none
list-style-image	url(/pix/bille.gif)
list-style-position	inside | outside

```
<html>

<style type="text/css">
```

```
<!--
.test1 {list-style-type: square;}
.test2 {white-space : pre; list-style-type: circle;}
.test3 {list-style-type:  lower-roman;}
-->
</style>

<body>

<table width="95%" border="1">

<tr><td>
<ul class="test1">
<li class="test2">Maître
      Corbeau, sur un arbre perché, </li>
<li>Tenait          en son bec un fromage. </li>
</ul>
</td></tr>

<tr><td>
<ul class="test3">
<li>Maître           Corbeau, sur un arbre perché, </li>
<li>Tenait          en son bec un fromage. </li>
</ul>
</td></tr>

</table>

</body>
</html>
```

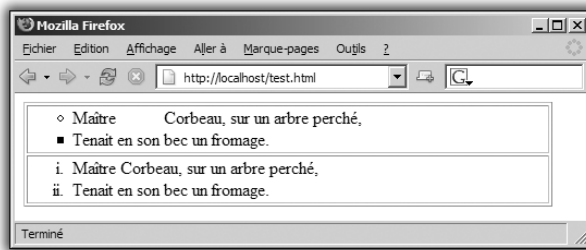


Figure 19.8 : *Exemple de listes*

Chapitre 20

Index

Index des fonctions

A

abs ... 530
addslashes ... 277, 538, 550, 616
array ... 116, 561
array_combine ... 562
array_count_values ... 562
array_diff ... 562
array_fill_keys ... 563
array_filter ... 563
array_flip ... 564
array_intersect ... 564
array_keys ... 564
array_map ... 565
array_merge ... 565
array_merge_recursive ... 566
array_pad ... 566
array_pop ... 567
array_push ... 567
array_rand ... 567
array_reduce ... 568
array_reverse ... 568
array_search ... 577
array_shift ... 568
array_slice ... 569
array_splice ... 569
array_sum ... 570
array_unique ... 571
array_unshift ... 571
array_values ... 571
array_walk ... 572
arsort ... 573
asort ... 573

B-C

basename ... 588
base_convert ... 531
bin2hex ... 538
bindec ... 531
ceil ... 531
chdir ... 589
checkdate ... 583
chgrp ... 589
chmod ... 589
chop ... 539
chown ... 590
chr ... 539
chunk_split ... 539
clearstatcache ... 590
closedir ... 590
compact ... 573
constant ... 645
copy ... 590
cos ... 531
count ... 220, 574
count_chars ... 540
crc32 ... 540
crypt ... 541
current ... 574

D

date ... 583
decbin ... 532
dechex ... 532
decoct ... 532
define ... 646
defined ... 646
deg2rad ... 532
delete ... 591
die ... 217
dirname ... 591
diskfreespace ... 592
disk_free_space ... 592
disk_total_space ... 592
dl ... 642

E

echo ... 541
empty ... 216, 638
end ... 574
ereg ... 226
error_log ... 499
eval ... 646
exit ... 217, 646
exp ... 533
explode ... 541
extract ... 576

F

fclose 592
feof 592
fflush 593
fgetc 593
fgetcsv 593
fgets 594, 605
file 594
fileatime 595
filegroup 595
fileinode 595
fileowner 596
fileperms 596
filesize 344, 596
filetype 596
file_exists 594
flock 597
floor 533
fopen 342, 597, 605
fpassthru 598
fread 344, 598
fscanf 599
fseek 599
fstat 600, 609
ftell 601
ftruncate 601
fwrite 601, 605

G

getcwd 601
getdate 584
getElementById 205
getenv 642
getimagesize 623
getmygid 643
getrandmax 533
gettype 638
get_cfg_var 642
get_current_user 643
get_defined_constants 643
get_extension_funcs 643
get_loaded_extensions 644
get_magic_quotes_gpc 644
get_meta_tags 542
gmdate 584
gmmktime 585
gmstrftime 587

H-I

header 188, 355
hexdec 533
highlight_file 647
highlight_string 647
htmlentities 543
htmlspecialchars 543
Image2WBMP 623
ImageAlphaBlending 623
ImageArc 624
ImageChar 625
ImageCharUp 625
ImageColorAllocate 626
ImageColorAt 627
ImageColorClosest 627
ImageColorDeAllocate 627
ImageColorExact 627
ImageColorsTotal 628
ImageColorTransparent 628
ImageCopy 628
ImageCreate 628
ImageCreateFromGIF 629
ImageCreateFromJPEG 629
ImageCreateFromPNG 629
ImageCreateFromString 629
ImageCreateFromWBMP 629
ImageCreateFromXBM 629
ImageCreateFromXPM 629
ImageCreateTruecolor 624
ImageDestroy 629
ImageFill 629
ImageFilledPolygon 629
ImageFilledRectangle 630
ImageFillToBorder 630
ImageFontHeight 631
ImageFontWidth 631
ImageGammaCorrect 627
ImageGIF 631
ImageInterlace 632
ImageJPEG 632
ImageLine 633
ImageLoadFont 633
ImagePaletteCopy 633
ImagePNG 632
ImagePolygon 633
ImageRectangle 634
ImageSetPixel 635
ImageString 635
ImageSX 635
ImageTTFBBox 635
ImageTTFText 636
ImageTypes 637
ImageWBMP 632
implode 543
ini_alter 644
ini_get 112, 644
ini_restore 645
ini_set 112
in_array 219, 577

include 109, 315, 453
isset 218, 639
is_array 220, 638
is_bool 638
is_dir 601
is_double 638
is_executable 602
is_file 602
is_float 638
is_int 638
is_integer 638
is_link 602
is_long 638
is_null 638
is_numeric 638
is_object 638
is_readable 602
is_real 638
is_resource 638
is_scalar 639
is_string 639
is_uploaded_file 280, 602
is_writable 602
is_writeable 602

J-K-L

join 544
JPEG2WBMP 637
key 578
krsort 578
ksort 578
link 602
list 579
log 534
log10 534
lstat 609
ltrim 559

M-N

mail 237
max 534
md5 544
microtime 585
mkdir 602
mktime 585
move_uploaded_file 280, 602
mt_rand 535
mysql_affected_rows 611
mysql_change_user 612
mysql_close 314, 612
mysql_connect 613
mysql_create_db 613
mysql_data_seek 614
mysql_db_name 614
mysql_db_query 615
mysql_drop_db 615
mysql_errno 615
mysql_error 615
mysql_escape_string 616
mysql_fetch_array 616
mysql_fetch_assoc 616
mysql_fetch_field 617
mysql_fetch_object 618
mysql_fetch_rox 616
mysql_free_result 618
mysql_get_client_info 622
mysql_get_host_info 622
mysql_get_proto_info 622
mysql_get_server_info 622
mysql_insert_id 619
mysql_list_dbs 614, 619
mysql_list_fields 619
mysql_list_tables 620
mysql_num_fields 620
mysql_num_rows 620
mysql_pconnect 613
mysql_query 615, 621
mysql_result 621
mysql_select_db 615, 621
natsort 579
next 574
nl2br 453, 544
number_format 535

O-P

ob_end_clean 518
ob_get_contents 518
ob_start 518
octdec 536
opendir 603
ord 544
parse_ini_file 604
parse_str 545
pathinfo 605
pclose 605
phpinfo 69, 528
phpversion 645
pi 536
popen 605
pos 575
pow 536
preg_match 222, 560
preg_replace 560

prev 574
print 545
printf 545
print_r 639
putenv 645

Q-R

quotemeta 547
rand 537
range 580
readdir 605
readfile 598, 606
readlink 606
realpath 609
rename 347, 606
require 516
reset 574
rewind 607
rewinddir 607
rmdir 607
round 537
rsort 581
rtrim 559

S

serialize 134, 393
session_start 472
setcookie 442
settype 640
set_file_buffer 608
shuffle 580
sizeof 574
sleep 648
sort 581
split 461, 560
sqrt 538
srand 537
sscanf 547, 599
stat 609
strchr 554
strcmp 548
strcspn 549
strftime 585
stripslashes 550
strip_tags 550
stristr 554
strlen 218, 551
strpbrk 551
strpos 552
strrchr 553
strrev 553
strrpos 553
strspn 554
strstr 554
strtolower 554
strtotime 588
strtoupper 555
strtr 556
str_pad 551
str_repeat 553
str_replace 453, 555
str_split 556
substr 556
substr_compare 557
substr_count 557
substr_replace 557
symlink 610

T

tempnam 610
time 587
tmpfile 610
touch 610
trim 217, 558

U-V-W

uasort 582
ucfisrst 555
ucwords 555
uksort 582
umask 611
uniqid 648
unlink 347, 591, 611
unserialize 135
unset 640
urlencode 175
usleep 648
var_dump 387, 641
wordwrap 559

Index

!

% 81
* 81
+ 81
.= 84
/ 81
$_COOKIE 442
$_GET 168
$_POST 179
$_REQUEST 180
$_SERVER 194
$_SERVER['PHP_SELF'] 452
$_SESSION 472

A

ActionScript 45
Addslashes 277
Adresse IP 665
Affectation 74
Ajax 226
ALTER 277
Animations Flash 41
Antislash 83, 538
Apache 23, 658
Appellation des variables 85
Applet Java 41
Application cliente 32
ASP 24
Attaque XSS 494
Attribut 411
Authentification 192, 292
Auto-incrémentation 264
Autocomplétion 525
Avantage
  rapidité 20
  scalabilité 20
  sécurité 20
  stabilité 20

B

Back-office 292
Balises PHP 67
Base de données 250
  connexion 262
  création d'une table 256
  MySQL 251, 254, 611
  relationnelle 371
Binaire 17
Blancs 539
Bloc de code 67
Blog 654
Boucle 94
Break 97, 519
Bzip2 355

C

Cache 341
Caractère d'échappement 83
Cascading Style Sheet (CSS) 326
Case 93
Casse 38, 78
Chaîne de caractères 80, 538
Changement de type 85
Classer un tableau 578
Clé 254
Clé primaire (SQL) 258
Client-serveur 35
Code
  ASCII 89, 548
  source 14, 18
ColdFusion 25
Colonne 254
Commentaire
  HTML 73
  PHP 72
Comparaison de chaînes 548
Compilateur 17
  PHP 19
Compression 352
Concurrents 23
Configuration PHP 60
Connexion à une base de données 262
Console JavaScript 210
Constante 78, 643, 668
  mathématiques 529
Continue 97, 519
Cookie 440
  date d'expiration 444
  de session 444, 480
Count 220
Création d'une table 256
Cryptage 541
CSS 326, 455

D

Dates et heures 140
Décompression 352
DELETE 308
Descripteur de fichier 342
Détection de navigateur 663
Die 217
Display_errors 497
DNS 36
Do ... while 95
Document.getElementById 205
Documentation PHP 650

E

Each 575
EasyPHP 652
Échappement de caractère 276
Éditeur PHP 59
Emacs 652
Empty 216
En-tête HTTP 353
Encodage de données 175
Enregistrement 254
Envoi
  de courriels 237
  de fichier 278
Ereg 226
Error_log 499
Error_reporting 498
Exit 217
Expression régulière 222, 560
Extension 22, 642
  de fichier 40
  PHP 358

F

Fichier
  copie 340
  de log de PHP 499
  déplacement 340
  écriture 340, 343
  lecture 340, 344
  renommer 340
  suppression 340
Firebug 210
Firefox 653
Flash 358
Fonction (voir Index des fonctions) 68
  arguments 99
  MySQL 674
  paramètre 99
  récursivité 104
  trigonométriques 532
For 96
Foreach 121
Format
  CSV 356
  EXCEL 356
  GIF 366
  JPEG 365
  PDF 363
  PNG 366
  SGML 418
  SWF 358
  WBMP 366
  XML 418
Formulaire HTML 156
Fournisseur d'accès 35
Free software 22
Front-office 292

G

GD (bibliothèque) 365
Génération d'image 365
Gestion des dates 583
Graphique 376
Guillemets 82
GZIP 355

H

Header 308
Hébergeur 66
Here printing 507
HTML et PHP 72
HTTP 31
Httpd.conf 64

I

I18N 29
Identifiant de session 480
If else 87
  elseif 92
  raccourcis 507
Image 365
Include 500, 516
Inclure un fichier 109, 315

Incrémentation ... 77
Index (SQL) ... 258
Input ... 159
  button ... 165
  checkbox ... 164
  hidden ... 166, 300
  password ... 160
  radio ... 164
INSERT INTO ... 263
Installation ... 48
Interpréteur ... 17, 40
In_array ... 219
IP ... 36
Isset ... 218
Is_array ... 220
Is_uploaded_file ... 280

J

Java ... 43
Javascript ... 41, 198
  console ... 210
JSON ... 229

L

Lamp ... 30
Langage
  compilé ... 16
  de programmation ... 14
  interprété ... 16
Liaison entre tables ... 372
Librairie GD ... 365
Ligne aléatoire dans une table ... 523
LIKE ... 334
LIMIT ... 336
Limiter le nombre de résultats ... 336
Linux ... 23
Localhost ... 66

M

Magic quotes ... 276
  Magic_quotes_gpc ... 497
  Magic_quotes_runtime ... 497
  Magic_quotes_sybase ... 497
Mail ... 489
  HTML ... 242
Manipulation de fichiers ... 340, 588
Méthode ... 411
  get ... 167
  post ... 178
Ming ... 363
Modes d'ouverture d'un fichier ... 343
Move_uploaded_file ... 280
MySQL ... 251, 254, 611
  types de données ... 255
  mysql_close ... 314

N

Navigateur ... 35
  détection ... 663
Nom
  de domaine ... 36
  des variable ... 74
Nombre
  aléatoire ... 535
  entier ... 80
  flottant ... 80
  variable de paramètres ... 511

O

Objets ... 384
  attributs ... 385
  classe ... 385
  constantes ... 390
  constructeur ... 391
  destructeur ... 392
  exceptions ... 405
  héritage ... 398
  instanceof ... 388
  interfaces ... 401
  méthodes ... 385
  objets ... 387
  parent ... 398
  réflexion ... 409
Ob_end_clean ... 518
Ob_get_contents ... 518
Ob_start ... 518
Open source ... 21
Opérateur
  ++ ... 77
  = ... 76
  de comparaison ... 87, 513
  de concaténation ... 84
  logique ... 90
  mathématiques ... 81
Optimisation ... 341, 515
OPTIMIZE TABLE ... 524

ORDER BY 335
Output buffering 518

P

Page d'accueil 56
Paramètres
  de fonction par défaut 103, 510
  PHP 60
Perl 26
Php.ini 60, 112, 236
Phpinfo 69
PhpMyAdmin 56, 283
PHP_SELF 453
Portabilité 18
PostgreSQL 255, 371
Preg_match 222
Préremplir un formulaire 299
Priorité 91
  des opérateurs 81, 656
Protocole 31
Prototype 214
Python 23

Q

Query string 175

R

Rafraîchir une page 526
Recherche 331
Référence 124
Regexp 222
Register_globals 497
Regular expression 222
Requête SQL 251
Require 516
Reset 166
Ruby 23

S

Select 162, 265, 267
Servlets 45
Session 472
  Session_start 472
SGBD 250
  SGBDR 371
Short tags 509
SimpleXML 421
Site officiel 650
SQL 254
SSL 503
String 80, 84, 538
Strlen 218
Structure de contrôle 86
Submit 166
Super-global 169
Switch 93
Syntaxe 66

T

Table 253
  de hachage 529
Tableau 116, 561
  affichage 126
  associatif 118
  ajouter un élément 116
  clé 118
  comparaison 137
  conversion de chaînes 128
  création 116
  index 117
  modifier un élément 117
  multidimensionnel 119
  opérateurs 136
  parcours 121
  scalaire 117
  taille 127
  tri 131
  union 136
TCP/IP 34
Template 347
Temps GMT 584
Textarea 161
Timestamp 140, 583
Transfert de fichier 667
Transtypage 85
Tri 331
Trim 217
Type
  array 116
  booléen 82
  de données 80
  de données (MySQL) 255
  de variable 638
  numérique 80
  transtypage 85

U

UPDATE ... 300
Upload ... 667
URL ... 35
Urlencode ... 175

V

Variable ... 74, 638
  $_GET ... 168
  $_POST ... 179
  $_REQUEST ... 180
  d'environnement ... 642, 668
  de configuration ... 642
  globale ... 107
  locale ... 107
  prédéfinie ... 657
  temporaire ... 515
Virtual host ... 659

W

Wamp Server ... 48, 651
Web ... 31
WHILE ... 94
Widgets ... 158
Wrapper ... 348

X

XHTML ... 161
XMLHttpRequest ... 226

Z

Zend ... 29
Zend Encoder ... 29
Zend Engine ... 28

Composé en France par Jouve
11, bd de Sébastopol - 75001 Paris